



Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations

Graduate School

2016

Interactomics-Based Functional Analysis: Using Interaction Conservation To Probe Bacterial Protein Functions

J. Harry Caufield
Virginia Commonwealth University

Follow this and additional works at: <https://scholarscompass.vcu.edu/etd>



Part of the [Bioinformatics Commons](#), and the [Other Microbiology Commons](#)

Downloaded from

<https://scholarscompass.vcu.edu/etd/4580>

This Dissertation is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

**INTERACTOMICS-BASED FUNCTIONAL ANALYSIS: USING INTERACTION
CONSERVATION TO PROBE BACTERIAL PROTEIN FUNCTIONS**

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at Virginia Commonwealth University.

by

J. Harry Caufield

Master of Science, Virginia Commonwealth University, 2012

Bachelor of Science, University of Delaware, 2008

Director: Peter H. Uetz, PhD

Associate Professor, Center for the Study of Biological Complexity,
VCU Life Sciences

Virginia Commonwealth University
Richmond, Virginia
December, 2016

ACKNOWLEDGEMENTS

I would like to thank my parents, my sister, my extended family, Danielle, my friends, the University, my advisor, my lab, my committee, and millions of generations of mammalian evolution for their assistance with this work and my well-being.

TABLE OF CONTENTS

	Page Number
List of Tables.....	v
List of Figures.....	vii
List of Abbreviations.....	ix
Abstract.....	1
1. Introduction.....	3
1.1 Background.....	3
1.1.1 The challenge of finding protein function.....	3
1.1.2 Interactomics as an approach to defining protein function.....	5
1.1.3 Protein complexes: the stable products of protein interactions.....	11
1.1.4 Interactomics of bacteriophages and their hosts.....	14
1.2 Research objectives.....	15
1.2.1 Addressing protein function through protein interaction networks.....	15
1.2.2 Measuring the extent of protein complex conservation.....	17
1.2.3 Using a meta-interactome to find commonalities between interactomes.....	18
1.2.4 Building a set of phage-host interactions to compare viral proteins.....	18
1.3 Project design and rationale.....	19
1.4 Intellectual merit.....	24
1.5 Broader impact.....	25
2. Conservation of Proteins in Bacterial Protein Complexes.....	28
2.1 Abstract.....	28
2.2 Introduction.....	29
2.2.1 The challenge of plentiful protein interaction data.....	29
2.2.2 Extending interaction analysis across species.....	32
2.3 Experimental methods.....	35
2.3.1 Scripts.....	35
2.3.2 Genome and complex data sources.....	36
2.3.3 Orthologous groups.....	37
2.3.4 Comparative proteome and complexome analysis.....	38
2.3.5 Protein complex interaction network assembly.....	46
2.4 Results and discussion.....	46
2.4.1 Conservation of proteins across bacterial genomes.....	46
2.4.2 The protein complexomes of <i>E. coli</i> and <i>Mycoplasma pneumoniae</i>	52
2.4.3 Using protein complexomes to predict complexes conserved in other species.....	57
2.4.4 Protein complexes and their essentiality are poorly conserved in bacteria....	63
2.4.5 The <i>E. coli</i> protein complexome as a model for other species.....	74
2.4.6 Essentiality of proteins in complexes and the impact of paralogy.....	81

2.4.7 Proteins of unknown function.....	89
2.4.8 Flexibility of protein complexes.....	94
2.4.9 Further discussion.....	102
3. Conservation of Protein-Protein Interactions among Bacteria.....	106
3.1 Abstract.....	106
3.2 Introduction.....	107
3.3 Experimental methods.....	113
3.3.1 PPI detection assay comparisons.....	113
3.3.2 Literature mining for citation analysis.....	115
3.3.3 Protein interaction data sets.....	116
3.3.4 Construction of meta-interactome networks.....	117
3.3.5 Interactome size prediction.....	120
3.3.6 Functional annotation.....	121
3.4 Results and discussion.....	122
3.4.1 The bacterial meta-interactome resembles individual interactomes in structure	122
3.4.2 Functional annotation of orthologous groups and their conservation.....	129
3.4.3 The meta-interactome predicts interactomes and their size.....	134
3.4.4 Biological differences vs. technical differences in interactomes.....	138
3.4.5 Meta-interactomes reveal broadly-conserved interactions involving proteins of unknown function.....	140
3.4.6 Interactomes are impacted by high-throughput experimental methods.....	145
3.4.7 Further discussion.....	152
4. Assessing Bacterial Protein Function using Bacteriophage Proteins.....	155
4.1 Abstract.....	155
4.2 Introduction.....	156
4.2.1 Microbiology in the context of bacteriophage interactions.....	156
4.2.2 Extending interaction analysis to viral proteins.....	157
4.3 Experimental methods.....	158
4.3.1 Data curation and data set assembly.....	158
4.3.2 Mycobacteriophage Giles protein-protein interactome.....	161
4.3.3 Data analysis.....	164
4.4 Results and discussion.....	164
4.4.1 An example of phage protein interactions from Mycobacteriophage Giles...	164
4.4.2 A curated set of phage-host PPI.....	170
4.4.3 A network and meta-network analysis of phage-host PPI.....	174
4.4.4 Phage-host PPI involve broadly-conserved protein complex components...	178
4.4.5 Additional discussion and future work.....	184
5. Conclusions.....	186
5.1 Protein complexes are irregularly conserved across divergent bacterial species	186
5.2 A protein-protein meta-interactome provides context for conserved interactions	186

5.3 A curated set of phage-host protein interactions provides a starting point for phage-host interactome screens.....	187
5.3 Future work.....	187
References.....	190
Vita.....	204
APPENDIX I. Guide to <i>spicednog</i>	205
I.I. User's guide to <i>spicednog</i>	205
I.I.I Setup.....	205
I.I.II Running <i>spicednog</i>	206
I.I.III Running accessory scripts.....	207
I.II. Code.....	209
I.II.I <i>spicednog.py</i>	209
I.II.II <i>spicednog-convert.py</i>	214
I.II.III <i>spicednog-marshmallow.py</i>	216
I.II.IV <i>ConToComplexCon.py</i>	218
APPENDIX II. Guide to <i>network_umbra</i>	220
II.I. User's guide to <i>network_umbra</i>	220
II.I.I Setup.....	220
II.I.II Running <i>Network_umbra</i>	221
II.II. Code.....	226
II.II.I <i>Network_umbra.py</i>	226
II.II.II <i>proteins_umbra.py</i>	265
APPENDIX III. Additional data tables for Chapter 2.....	283
APPENDIX IV. Additional data tables for Chapter 3.....	305
APPENDIX V. Additional data table for Chapter 4.....	311

TABLES

	Page Number
1-A. A selection of published, comprehensive protein-protein interactomes.....	8
2-A. Core set of bacterial species and strains with published essentiality screen results.	37
2-B. List of general functional categories used to describe protein complex function....	45
2-C. Hu et al. (2009) E. coli protein complexes and the best matches among EcoCyc E. coli protein complexes.....	79
3-A. Experimental microbial interactome sizes.....	107
3-B. Set of comprehensive bacterial protein interactome studies used for citation analysis.....	115
3-C. Reference proteomes used for interactome size prediction.....	121
3-D. Functional categories used to describe orthologous groups of bacterial proteins.	121
3-E. Predicted bacterial interactome sizes.....	135
3-F. Conserved interactions involving selected OGs of unclear function.....	141
4-A. Descriptive statistics of the phage-host protein interaction data set.....	171
APPENDIX III-A. Average conservation of loci and orthologous groups across numerous bacterial species.....	283
APPENDIX III-B. Average conservation of orthologous groups among protein complex components.....	284
APPENDIX III-C. Conservation of orthologous groups and protein-coding loci between pairs of model bacterial species.....	285
APPENDIX III-D. Key to short protein complex IDs.....	286
APPENDIX III-E. Conservation of E. coli complexes from Hu et al. (2009).....	290
APPENDIX III-F. Essentiality of E. coli complexes from Hu et al. (2009).....	291
APPENDIX III-G. Conservation of E. coli complexes from EcoCyc.....	292
APPENDIX III-H. Essentiality of E. coli complexes from EcoCyc.....	293
APPENDIX III-I. Conservation of Mycoplasma pneumoniae complexes from Kühner et al. (2009).....	294
APPENDIX III-J. Essentiality of Mycoplasma pneumoniae complexes from Kühner et al. (2009).....	295
APPENDIX III-K. Experimental protein complexes containing uncharacterized components.....	296
APPENDIX III-L. Complex-based protein-protein interactions for E. coli.....	304
APPENDIX IV-A. Counts of literature citing multiple bacterial interactomes.....	305
APPENDIX IV-B. All interactions in the meta-interactome network.....	308
APPENDIX IV-C. All interactions in the consensus meta-interactome network.....	309
APPENDIX IV-D. Conserved interactions of unclear function.....	310
APPENDIX V-A. Interactions between bacteriophage and bacterial proteins.....	311

APPENDIX V-B. Citations for phage-host protein interaction sources.....	313
---	-----

FIGURES

	Page Number
1-A. Concept of the interactome.....	6
1-B. Example interactomes from modeled and experimental data.....	10
1-C. Example interactomes from experimental data and interactors common between them.....	13
1-D. Flow chart of core elements of this study.....	22
2-A. Structure of the proteasome and interactions across five non-bacterial species.....	34
2-B. Example of fractional conservation value assignment.....	40
2-C. Example of protein complex size determination.....	42
2-D. Protein complexes are enriched for highly conserved components.....	48
2-E. Protein complex data sets vary in composition.....	54
2-F. Histogram of Kühner et al. <i>M. pneumoniae</i> complexes and average conservation fractions.....	58
2-G. Protein complex sets vary in conservation across bacteria.....	61
2-H. Examples of protein complex conservation across bacteria.....	65
2-I. Fractional essentiality and conservation of protein complexes across species.....	68
Image2.....	69
2-J. All EcoCyc complexes and their fractional conservation in selected bacterial species.....	69
2-K. All EcoCyc complexes and their fractional essentiality in selected bacterial species.....	71
2-L. <i>E. coli</i> complex conservation across Bacteria corresponds to taxonomic boundaries.....	76
2-M. <i>E. coli</i> experimentally-observed complex conservation across bacteria corresponds to taxonomic boundaries.....	80
2-N. Conserved complex components are enriched for essential proteins.....	83
2-O. Cross-species conservation of experimentally-observed protein complexes and the sums of the counts of potential paralogs of their components.....	86
2-P. Essentiality of proteins in complexes.....	88
2-Q. Protein complexes are rich in highly-conserved proteins of unknown function.....	90
2-R. Interaction network of <i>E. coli</i> protein complexes.....	93
2-S. All EcoCyc complexes and their fractional conservation in selected strains of <i>E. coli</i> and <i>Shigella</i>	98
2-T. Predicted protein complexes in <i>H. pylori</i>	101
3-A. Analysis of citations of bacterial protein interactome literature.....	110
3-B. Concept and construction of the meta-interactome.....	119
3-C. Composition of the meta-interactome.....	125
3-D. Overall structure of the main component of the consensus meta-interactome.....	126

3-E. Properties of the consensus meta-interactome.....	127
3-F. Conserved interactions in the consensus meta-interactome.....	131
3-G. Cross-functional interactions in the consensus meta-interactome.....	133
3-H. Predictions of maximal interactome size.....	137
3-I. The NDH-1 complex as an example of conserved interactions.....	144
3-J. Composition of the meta-interactome by interaction detection method.....	147
3-K. A comparison of high-throughput yeast two hybrid screens.....	149
4-A. The mycobacteriophage Giles interactome.....	166
4-B. Composition of the observed phage vs. host protein-protein interactions by phage or host.....	173
4-C. The network of phage vs. host protein-protein interactions.....	175
4-D. The meta-network of multiple-incidence phage vs. host protein-protein interactions.	177
4-E. The meta-network of phage protein vs. host complex interactions.....	181
A1. The initial spicednog prompt.....	206
A2. The results of a spicednog search.....	206
A3. The results of an example spicednog-convert.py search.....	208
A4. The results of an example spicednog-marshmallow.py search.....	208
A5. The network-umbra menu seen after meta-interactome construction.....	221
A6. Using network-umbra to retrieve a proteome.....	224
A7. Example of interactome prediction process.....	225

ABBREVIATIONS

3-AT	3-Amino-1,2,4-triazole
CRISPR	Clustered Regularly Interspaced Short Palindromic Repeats
DNA	Deoxyribonucleic acid
Mb	Megabase (one million base pairs) of DNA
OG	Orthologous Group
ORF	Open Reading Frame
PPI	Protein-Protein Interaction
PUF	Protein of Unknown Function
RNA	Ribonucleic acid
Y2H	Yeast Two Hybrid

ABSTRACT

INTERACTOMICS-BASED FUNCTIONAL ANALYSIS: USING INTERACTION CONSERVATION TO PROBE BACTERIAL PROTEIN FUNCTIONS

By J. Harry Caufield, M.S.

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor
of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2016.

Major Director: Peter H. Uetz, PhD
Associate Professor, Center for the Study of Biological Complexity,
VCU Life Sciences

The emergence of genomics as a discrete field of biology has changed humanity's understanding of our relationship with bacteria. Sequencing the genome of each newly-discovered bacterial species can reveal novel gene sequences, though the genome may contain genes coding for hundreds or thousands of proteins of unknown function (PUFs). In some cases, these coding sequences appear to be conserved across nearly all bacteria. Exploring the functional roles of these cases ideally requires an integrative, cross-species approach involving not only gene sequences but knowledge of interactions among their products. Protein interactions, studied at genome scale, extend genomics into the field of interactomics. I have employed novel computational methods to provide context for bacterial PUFs and to leverage the rich genomic, proteomic, and interactomic data available for hundreds of bacterial species.

The methods employed in this study began with sets of protein complexes. I initially hypothesized that, if protein interactions reveal protein functions and interactions are

frequently conserved through protein complexes, then conserved protein functions should be revealed through the extent of conservation of protein complexes and their components. The subsequent analyses revealed how partial protein complex conservation may, unexpectedly, be the rule rather than the exception. Next, I expanded the analysis by combining sets of thousands of experimental protein-protein interactions. Progressing beyond the scope of protein complexes into interactions across full proteomes revealed novel evolutionary consistencies across bacteria but also exposed deficiencies among interactomics-based approaches. I have concluded this study with an expansion beyond bacterial protein interactions and into those involving bacteriophage-encoded proteins.

This work concerns emergent evolutionary properties among bacterial proteins. It is primarily intended to serve as a resource for microbiologists but is relevant to any research into evolutionary biology. As microbiomes and their occupants become increasingly critical to human health, similar approaches may become increasingly necessary.

DISSERTATION

Chapter 1 – Introduction

1.1 Background

1.1.1 The challenge of finding protein function

Bacteria are ubiquitous, diverse, and constantly subject to genetic flux at a colossal scope. Despite more than 150 years of research since the emergence of modern microbiology, many of the intricate components and processes within even well-studied bacterial species remain mysterious. Even *E. coli*, the microbiologist's workhorse, contains a genome with more than a thousand open reading frames of unclear function. Identifying the roles, regulation, and interactions of microbial proteins depends upon a combination of comparison to known, conserved phenomena and experimental analysis. The latter approach has not been able to keep pace with the constant influx of genome and proteome data: fewer than 1% of protein sequences have functional annotations from experimental results (Erdin et al. 2011). The NCBI RefSeq database contained more than 50 million entries for bacterial proteins alone as of June 2016, suggesting that the functions of more than 49 million proteins have not been experimentally determined or even investigated.

The issue of defining protein function is not simply an issue of staggeringly large numbers. Meaningful associations between proteins may only emerge once proteins are

placed in their functional context, though this context may include dozens of other proteins. Recent developments in genome sequencing and high-throughput proteomics have been employed to address this issue. Additionally, a full understanding of proteins of unknown function must be considered within the context of other species. *E. coli* serves as an ideal model organism but provides just one, isolated genetic background. The roles of many proteins may only emerge once microbiomes and cross-species interactions are included in analyses. Confronting this issue therefore requires a philosophical shift. We must unify methodological advancements with holistic, data-aggregating approaches if we wish to illuminate the darker corners of bacterial proteomes.

Unclear protein function is not a purely research-based concern. Numerous recent microbiological studies have highlighted the tenuous relationships between humans and the bacteria in our environment and have revealed how otherwise uncharacterized proteins and cross-species interactions impact these relationships. Work by Zipperer et al. (2016) showed that a commensal bacterial species found in the human nose, *Staphylococcus lugenensis*, releases a peptide which can prevent colonization by pathogenic *Staphylococcus aureus* and *Enterococcus* strains. A multi-faceted analysis by Kamran et al. (2016) identified novel cell division-related proteins in the common human pathogen *Helicobacter pylori*. A study by Wu et al. (2016) identified a previously-unknown magnesium uptake protein and virulence factor in the virulent human pathogen *Francisella tularensis*. These are just three examples of how a better

understanding of bacterial protein function – and especially a focus on proteins of unknown function – can enhance understanding of common bacterial pathogens.

1.1.2 Interactomics as an approach to defining protein function

In this study, I leverage plentiful interactomics results using bioinformatics methods to re-interpret the data and draw new conclusions about bacterial genomes and proteins. Interactomics refers to the study of interactomes, where a single interactome is a further level of complexity beyond a genome or proteome: while a genome is the set of all genes in a genome and a proteome is the set of all proteins, an interactome is the set of all protein-protein interactions (PPIs) among the members of a proteome. Interactomes are frequently visualized and interpreted as graphs (**Fig. 1-A**). Like genomes and proteomes, interactomes may be conceptually complete but functionally incomplete. A protein present in nature but omitted from experimental observations of a proteome will in turn be missing from an interactome and experimental screens performed to build an interactome nearly always fail to detect some fraction of the true interactome. Furthermore, the conceptual interactome is an abstraction, as the significance of a PPI may vary between pairs or complexes of proteins. Even so, interactomics is a crucial element of a modern, systems biology approach to microbiology and provides rich context for informing protein function.

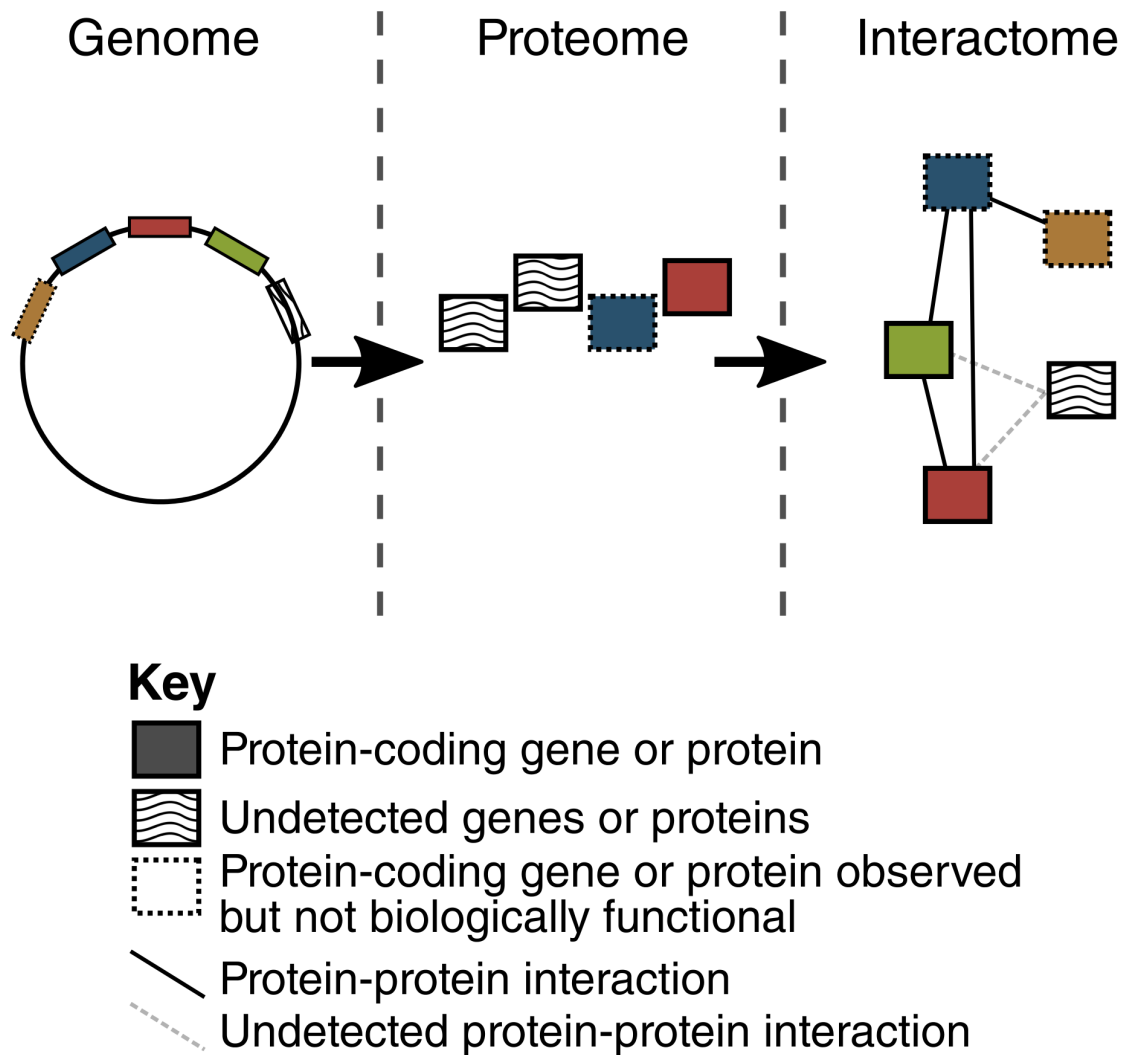


Fig. 1-A. Concept of the interactome. Each genome (shown here as a circle, with open reading frames shown as colored boxes) codes for proteins that contribute to a proteome. The proteome, in turn, is the source of the interactors in an interactome, as the interactome defines the set of interactions among proteome members. All three concepts shown here are abstractions and may be incomplete for a combination of biological and methodological reasons, e.g., a protein-coding sequence may exist in a genome but may escape annotation or its product may not be detected in a purified proteome. As binary protein-protein interactomes are usually determined using cloned open reading frames rather than purified proteins, a protein not found in an experimentally-defined proteome may still be included in an interactome, as is shown here.

Complete interactomes have been defined for a handful of species, most of them microbial (**Table 1-A**). This table is not intended to be a complete listing of all interactomes as this remains an active field. As shown in **Fig. 1-A**, a complete experimental interactome is always some fraction of the true biological interactome for a combination of natural and methodological reasons. Even a well-designed study of numerous protein-protein interactions may therefore not be comprehensive if researchers intentionally omit potential interactors. Some researchers have used the term “interactome” to refer to other large sets of interaction screens, such as a study of *C. elegans* proteins by Li et al. (2004). This study focused on 3,024 protein-coding genes out of an estimated 17,800, or only ~17% genome coverage and likely less coverage of the *C. elegans* proteome. In this work, I use the term “comprehensive” to refer only to studies with at least ~60% proteome coverage.

Sets of microbial proteins serve as ideal subjects for interactomics studies: their proteomes generally contain only a few thousand proteins rather than the potentially more than 20 thousand proteins in the human proteome (Kim et al. 2014) without counting protein variants or post-translational modifications. Single-celled organisms also offer the benefit of comparatively few cell compartments; interactomes in higher organisms may only make sense for each cell type. Comprehensive interactomes have been published for more than nine microbial species, and in some cases, researchers have also identified comprehensive sets of protein complexes (or, more simply, the stable products of protein-protein interactions).

Table 1-A. A selection of published, comprehensive protein-protein interactomes.

Taxonomic Superkingdom	Species	Proteins in Ref. Proteome	Complexes	Binary Interactions	Citation(s)
Bacteria	<i>Campylobacter jejuni</i>	1,623	-	11,687	Parrish et al. (2007)
	<i>E. coli</i>	4,305	310*	2,234**	*Hu et al. (2009); **Rajagopala et al. (2014)
	<i>Helicobacter pylori</i>	1,553	-	1,515	Häuser et al. (2014)
	<i>Mesorhizobium loti</i>	7,255	-	3,121	Shimoda et al. (2008)
	<i>Mycobacterium tuberculosis</i>	3,991	-	>8,000	Wang et al. (2010)
	<i>Mycoplasma pneumoniae</i>	687	259	-	Kühner et al. (2009)
	<i>Synechocystis spp.</i> PCC 6803	3,507	-	3,236	Sato et al. (2007)
	<i>Treponema pallidum</i>	1,028	-	3,649	Titz et al. (2008)
Eukaryota	<i>Saccharomyces cerevisiae</i>	~3,500 – 6,700	491 - 547	10,000 – 30,000	Uetz et al. (2000); Ito et al. (2001); Schwikowski et al. (2000); Gavin et al. (2005); Krogan et al. (2006)
	<i>Schizosaccharomyces pombe</i>	5,121	-	2,278	Vo et al. (2016)
Viruses	Bacteriophage lambda	66	-	97	Rajagopala et al. (2011)
	Bacteriophage Giles	77	-	137	Mehla et al. (2015)
	Hepatitis E virus	3 (processed into >11 fragments)	-	25	Osterman et al. (2015)

“Proteins in Ref. Proteome” refers to the number of proteins in the specified proteome as defined by Uniprot; individual studies may not include all proteins in their interaction screens.

It should be noted that the expected size of an interactome can be difficult to estimate based only on the interactors involved. In a very simple network such as that defined by a Barabási–Albert model (Albert and Barabási 2002) (**Fig. 1-B-A**), the overall structure may appear similar to that of a protein interaction network in that it has a scale-free degree distribution (**Fig. 1-B-C**). It fails to capture additional properties of an experimental protein interactome such as that of *E. coli* (**Fig. 1-B-B**). In comparing sets of protein interactions, we therefore must recognize that the overall data sets may follow similar trends in terms of degree distribution, but likely demonstrate novel properties at the local level.

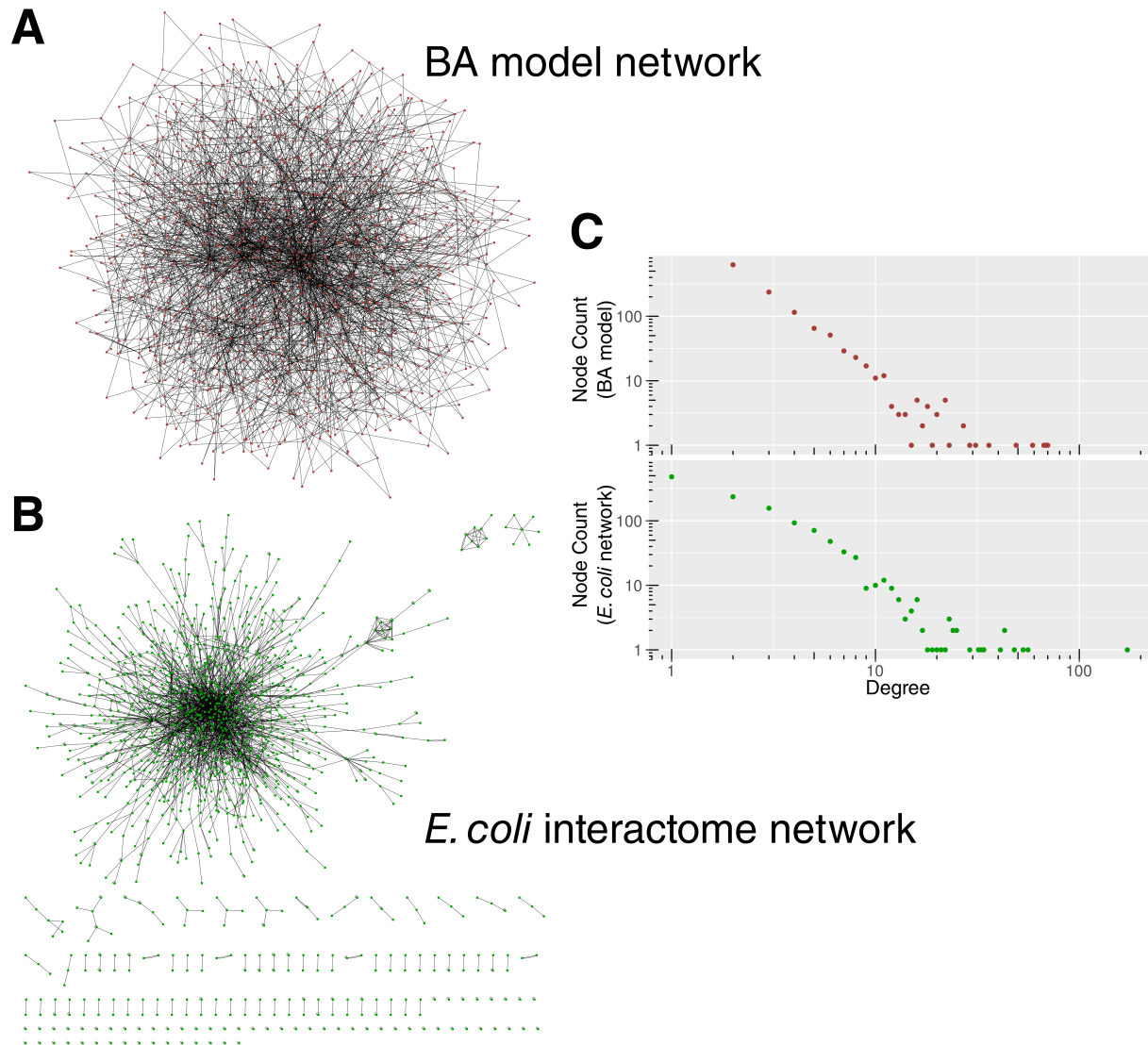


Fig. 1-B. Example interactomes from modeled and experimental data. **A)** A network of 1,230 nodes generated using a Barabasi-Albert model such that each node is connected to at least two other nodes. Generated with the Network Randomizer 1.1.1 plugin for Cytoscape 3.4.0. **B)** Network of *E. coli* protein-protein interactions as reported by Rajagopala et al. (2014). This network also contains 1,230, though in this network each node corresponds to an *E. coli* protein. **C)** Comparison of degree distributions for each network, with both axes on a log scale. Both degree distributions appear to follow a power law.

1.1.3 Protein complexes: the stable products of protein interactions

Protein complexes provide some of the most easily understood and most evolutionarily conserved examples of protein interactions. Nearly all of the enzymes most crucial to all forms of life are constructed from proteins, including RNA and DNA polymerases, chaperones like GroEL and Hsp60, the proteasome, the degradosome, and ATP synthases. Even the ribosome, with its substantial RNA content, depends upon the interactions between numerous proteins for stability. It is therefore critical to consider protein complexes when discussing comprehensive sets of PPIs. The most easily observable PPIs will generally be those participating in the most stable interactions and forming the most stable complexes.

Protein complexes contribute just one fraction of the total potential for PPI in any single organism. The general structure of some complexes is well-conserved, even when the exact sequences of the interacting complex components differ. We may then expect to see similar numbers of PPIs across different species, especially if their genomes code for broadly-conserved protein complexes, but also if they code for other well-conserved, interacting proteins. Even with cases of differing sequence, however, the lack of a given protein sequence in a proteome may also indicate lack of a corresponding protein complex component and hence lack of the PPIs involving that component.

As seen in **Table 1-A**, there does not appear to be a consistent relationship between proteome size and interactome size. Two different species (e.g., *C. jejuni* and *H. pylori*) may code for very similar counts of proteins in their respective proteomes but may differ in interactome sizes by thousands of interactions. These results appear surprising without the context of biology and methodology (though even with the context, the inconsistency is puzzling). Some proteins may have hundreds of interacting partners yet remain restricted to specific taxons, just as some phenotypes and behaviors are restricted to various branches of the tree of life (**Fig. 1-C**). Other proteins may appear to be conserved between species but may operate in different functional contexts and contribute different amounts of interactions in different species.

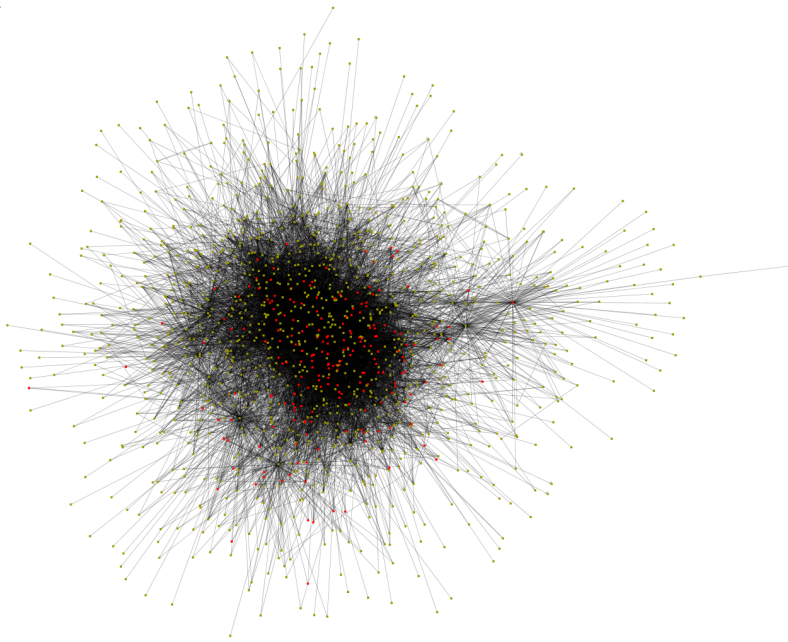
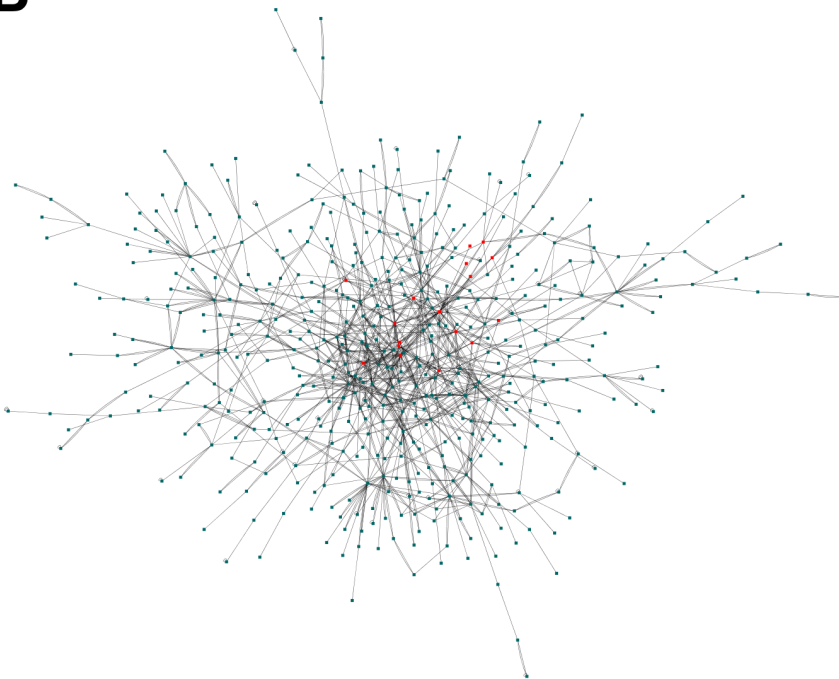
A**B**

Fig. 1-C. Example interactomes from experimental data and interactors common between them. A) The largest component of the *Campylobacter jejuni* interactome as published by Parrish et al. (2007), containing 1,307 nodes and 11,918 edges. The highest degree node in this network, with 208 edges, is the predicted histidine triad (hIT) protein Cj0499 (Uniprot: Q0PB14). All proteins interacting with Cj0499 are highlighted in red. **B)** The largest component of the *Helicobacter pylori* interactome as published by Häuser et al. (2014), containing 502 nodes and 1,263 edges. The closest ortholog to Cj0499 is HP_0741 (Uniprot: O25440), a protein coded for by an *H. pylori* gene but not present in this interactome. The highest degree interactor in this network, with 31 edges, is HP_1262 (Uniprot: O25852), a NADH-quinone oxidoreductase. All proteins interacting with HP_1262 are highlighted in red.

It is likely, however, that much of these differences are the result of methodological discrepancies. Researchers performing experimental interaction screens start with different proteins but often assume this set is representative of the entire proteome. They then apply different filtering procedures to the resulting interaction data based on internal considerations of confidence and relevance. In the *C. jejuni* interactome study published by Parrish et al. (2007), the authors used 1,477 ORFs (or 91 percent of the reference proteome, roughly) in their interaction screens, found 11,687 repeatable interactions in their screens, and filtered this set to 2,884 using measures of biological relevance (primarily, similarity to the *E. coli* and *H. pylori* interactomes available at the time). In comparison, the *H. pylori* interactome produced by Häuser et al. (2014) is the product of 1,587 ORFs from two different *H. pylori* strains which yielded a “raw” interactome of 2,154 PPIs, a filtered set of 1,515 and a final, “high quality” core set of 908 PPIs. In this study, the final filtering procedure was based on a threshold where the authors observed “a conspicuous increase of the prey count”. (Interaction screens frequently distinguish between the two halves of a binary interaction as bait and prey interactors.) A discussion of differences in interactomics methodologies continues in Chapter 3.

1.1.4 Interactomics of bacteriophages and their hosts

Interactomics studies have not been limited to cellular life: work has included analyses of viruses infecting humans and, more frequently, those infecting bacteria. These

viruses – the bacteriophages – are just as ubiquitous as bacteria, if not more so. Starting with a rough population estimate of 10^{30} bacterial cells on Earth, various estimates have suggested between an equivalent to a 100-fold greater population of bacteriophages (Wommack and Colwell 2000, Rowher 2003, Clokie et al. 2011). Phages serve as a massive and constant source of new genetic variation, both as the result of phage-mediated genetic transfer (that is, transduction) and through the perpetual battle between viruses and their hosts (Hambly and Suttle 2005, Hatfull and Hendrix 2011). For these reasons, it is likely that any discussion of bacterial interactomes is limited if it fails to consider interactions between viral and host proteins.

Bacteriophage interactome studies, including those of famous coliphage lambda (Rajagopala et al. 2011) and the *Streptococcus* phages Cp-1 (Häuser et al. 2011) and Dp-1 (Sabri et al. 2011), have provided attractive interactomics subjects due to the small viral proteome sizes and corresponding lower level of expected complexity. Protein-protein interactions between phage and their hosts are, with a few exceptions (Roucourt and Lavigne 2009, Blasche et al. 2013), largely unexplored. Due to the intimate relationships between bacteria and viruses, we may use these interactions as a novel venue for exploring bacterial genes of unknown function.

1.2 Research objectives

1.2.1 Addressing protein function through protein interaction networks

Interactomics is a promising field with the potential to revolutionize our understanding of protein functions and evolution. The results of interactomics approaches exemplify the core tenets of systems biology: rather than considering proteins as discrete entities with inherent properties driving their functions, interactomics considers each protein to be one participant in a full interactome. The functions of a particular protein are then dependent not only on the proteins and other materials it interacts with, but also upon the other participants in this interaction network. Our understanding of each of these interactome networks is always incomplete (**Fig. 1-A**). One of the primary challenges in working with any interactome is therefore addressing the limits of the data.

As part of the studies described here, I have developed novel bioinformatics-based approaches to interpreting protein-protein interactomes. Moreover, I have constructed ways to predict interactome size and composition for bacterial species with limited protein interaction data. I have extended my results beyond single-species interactomes into host vs. virus protein interactions, resulting in a unique, curated collection of these interactions. This project essentially defines a bacterial pan-interactome.

This work addresses a major conceptual gap in how the results of microbiological research are interpreted. Bacterial genomics studies provide a wealth of data: with 73,397 genome entries in NCBI GenBank as of September 2016 (though just 5,813 genomes, or about 8 percent, are annotated as “complete”), more species of bacteria have had their genomes sequenced than any other branch of the tree of life. Similarly,

as of September 2016, the Gene Ontology Consortium's AmiGO 2 database contains 458,273 different gene annotations for bacteria, yet 133,988 have no experimental evidence to support any functional role. Numerous other genes have annotations based on observations from distantly-related species. At this point, the overall question is quite simple: what is the most efficient way to obtain functional information for bacterial genes of unknown or unclear function? Furthermore, given a set of gene products and their functions for a single bacterial species, how useful are those products as a model for those encoded by the genomes of other bacterial species?

1.2.2 Measuring the extent of protein complex conservation

I have designed this project from the perspective that protein function is best understood using a protein's role within *multiple* sets of interactions between those proteins. This work applies such an approach across hundreds of different bacterial species. As noted above, protein complexes provide useful sets of protein interactions but provide just one part of the interactome of any species. To date, few studies have compared protein complexes across multiple species and none have taken a purely bacteria-centric perspective. Comparative interactomics studies have been performed with the eukaryotes yeast (*Saccharomyces cerevisiae*), nematode (*Caenorhabditis elegans*), fly (*Drosophila melanogaster*) and human (Gandhi et al 2006; Haynes et al 2006). In these cases, the similarities between interactomes were generally found to be small: Gandhi et al. (2006) found just 42 protein-protein interactions shared between the four species listed above.

Determining the conservation of bacterial protein complexes therefore allows these complexes to be used as controls in the comparison of interactomes. If a complex is expected to be conserved among a set of genomes, then its interactions are also expected to be conserved. Bacterial genomes are particularly useful in this context, as unlike the human interactome, several bacterial interactomes are as close as possible to complete (see **Table 1-A**).

1.2.3 Using a meta-interactome to find commonalities between interactomes

A cross-species approach not only permits observation of similarities among protein interactions and functions but also allows quantification of deficiencies in the available interaction data. Due to methodological differences and the inherent bias of growing bacteria in the lab rather than in the wild – among other factors – no experimental interactome can capture the full extent of biologically-relevant protein-protein interactions. This renders cross-species interactome comparison difficult, as an interaction found in one species but missing in another may be missing due to biological phenomena or methodological variation. This work establishes a model set of interactions (in the form of a meta-interactome; see Ch. 3) to provide predicted interactions for bacterial species with limited to no experimental interaction data.

1.2.4 Building a set of phage-host interactions to compare viral proteins

The penultimate chapter of this work demonstrates how a cross-species protein interaction comparison can be extended further. In this case, the goal is to quantify similarities among interactions between bacteriophage and bacterial proteins. Though the evolutionary history of bacteria as a whole has been and continues to be shaped by interactions with bacteriophages, few studies have identified comprehensive sets of interactions between these viruses and their hosts. Furthermore, little work has been done to observe patterns among virus vs. bacterial protein interactions. This work provides a database of interactions to serve as a resource for all researchers concerned with bacteriophage and bacterial evolution.

1.3 Project design and rationale

This project uses computational approaches to interpret under-utilized sequence and interaction data. A primarily bioinformatics-based approach is preferred in studies of protein-protein interactions for three primary reasons. First, much as in the field of genomics, the amount of data produced by even a single experimental study is so vast that its original authors may find a comprehensive analysis impractical (or, at least, outside the scope of the original study). Additionally, comparing data sets requires the development of new, flexible data analysis methods, occasionally because the researchers responsible for the experimental data designed their methods in a species-specific manner. A cross-species approach can therefore extend conclusions to species for which experimental work is not feasible, such as non-culturable or highly pathogenic bacteria. Finally, computational approaches yield resources which are directly

applicable to future work. The software and data sets produced during the course of this work are immediately usable for any researcher concerned with bacterial protein-protein interactions.

Fig. 1-D provides an overview of the strategy of this overall work and the initial data sets used in the process. All three of the primary focus areas of this work depend upon two primary types of data: protein-protein interactions (PPI) and orthologous groups. In the first segment of this work – as described in Chapter 2 – the PPI are abstracted from sets of protein complexes. These sets include a set of literature-defined complexes from the EcoCyc database (Keseler et al. 2013) and two sets of protein complexes defined directly from individual proteomes (Hu et al. 2009; Kühner et al. 2009). Each member of a protein complex is inferred to have the potential for an interaction with each other member of that complex. The PPI in the second two segments of this work are binary interactions primarily curated from the IntAct molecular interaction database (Orchard et al. 2014), chosen specifically for its large collection of bacterial PPI and its inclusion of interactions from several other relevant databases. In all areas of this project, the proteins involved in each interaction are compared across species by mapping each to an orthologous group (OG). OGs permit proteins to be clustered together by sequence similarity and enable taxonomy comparisons (e.g., we may find members of a particular OG in strains of *E. coli* but rarely in other species). In lieu of developing an entirely novel method for defining OGs, I elected to use eggNOG (Huerta-Cepas et al. 2016), as it defines OGs using specific taxonomic levels rather than by similarity with every other

known sequence. The most recent version of eggNOG also includes OG assignment for viral proteins, enabling cross-virus comparisons of virus vs. host PPI (specifically, those described in Chapter 4 of this work).

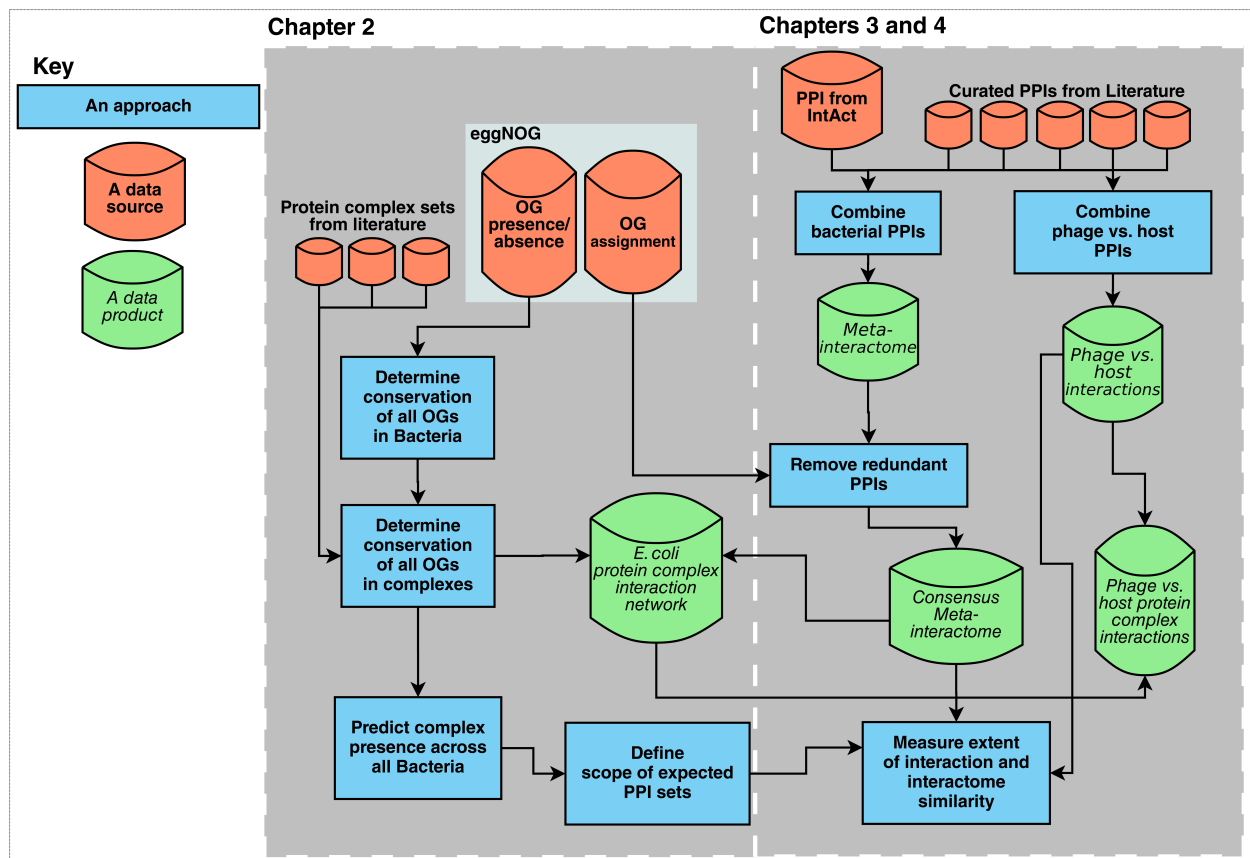


Fig. 1-D. Flow chart of core elements of this study.

The first aspect of this study, detailed in Chapter 2, focuses on conservation of protein complexes in bacteria. Protein complexes and their associated sequences (e.g., ribosomal RNA) are known to include the most broadly conserved sequences in bacterial genomes. Perhaps more importantly, protein complexes generally retain their structures and functions across species. This assumption is, of course, based on well-studied protein complexes such as ribosomes, polymerases, and chaperones. I therefore began this work with the expectation that the conservation of protein complexes would serve as an ideal proxy for the conservation of protein-protein interactions (PPIs). If a protein complex seen in one species is missing in another, for example, we may naturally predict that the PPIs seen among the complex components in the first species cannot be present in the second species. Conservation of protein complexes also provides a functional viewpoint into PPI conservation, as a complex with a missing component presumably either can no longer participate in functions necessitating that component or has gained a novel function.

Chapter 3 of this work concerns comparisons of experimental interaction data. While comprehensive interactomes have been published for more than a few bacterial species and beyond (see **Table 1-A**), it is difficult to quantify the similarities among data sets. These similarities are crucial to understanding biological phenomena in a broader evolutionary context. I elected to combine the available data in an orthology-dependent manner, forming a “meta-interactome” in which each interaction represents PPI from one or more data sets or species.

The final chapter of the work, Chapter 4, is a compilation of bacteriophage vs. host protein interactions. Thus far, few interactome studies have studied the PPIs involved in the course of a full bacteriophage life cycle. The process is poorly understood and likely varies significantly between viruses and hosts, yet some similarities must exist, especially in cases when the phage and host each contain proteins found to interact in a different pair of virus and host. The majority of the effort in this section is therefore based upon curation of PPIs observed in published experimental results. I then integrate these interactions using an orthology-based approach and network analysis to ascertain how consistent phage vs. host PPIs may be.

1.4 Intellectual merit

This study focuses on three concepts directly relevant to bacterial evolution: protein conservation, protein-protein interaction conservation, and the functions of uncharacterized proteins in the context of their interactions. The core idea behind all three of these aspects is that protein sequences are best understood using a cross-species approach. Furthermore, it is the pattern of a protein or interaction's conservation across species that offers the clearest viewpoint of its role or roles in different branches of bacterial taxonomy. Such an approach is made possible by the wealth of sequence and interaction data available at this time but differs from the single-protein or single-species approach traditionally employed by microbiologists.

The encoded PUFs and domains of unknown function (DUFs), despite their unexplained biological roles, are not simply dispensable. Many of the genes in these regions code for proteins essential to life in numerous bacterial species (Goodacre et al. 2013). If these genes are found to interact with phage proteins, we may reverse-engineer the interactions as novel methods for controlling bacterial pathogens.

This work is unique among computational biology studies in that I have included all of its resulting data tables, the code I used to produce the data, and guides to re-using the code. It is unfortunately not yet common practice to provide all three of these features. It is my hope that these contributions will maximize this work's potential for long-term re-use.

1.5 Broader impact

Bacteria and their proteins pose a quandary for human health. Growing resistance to the usual antibiotic therapies threatens thousands of lives each year; as of 2013, approximately 23,000 deaths could be directly connected to antibiotic-resistant infections each year in the United States alone (Centers for Disease Control and Prevention 2013). *Staphylococcus aureus*, the canonical example of antibiotic resistant pathogenesis, has been evading antibiotic treatments since at least the 1950's (Shanson 1981) and is associated with between 60,000 to 320,000 deaths each year in the United States (van Hal et al. 2012). Antibiotic resistance also has the potential to allow treatable infections such as those from foodborne *Salmonella* (Helms et al. 2004;

Helke et al. 2016), opportunistic *Klebsiella pneumoniae* (Holt et al. 2015) to become chronic diseases with dangerous consequences. Even when infections are not life-threatening, antibiotic resistance can increase their duration, as has been observed with ear infections (Sillanpää et al. 2016). We have also found that a scorched-earth, all-or-nothing approach to antibacterials is ineffective and often lethal: eliminating bacterial pathogens in human patients with broad-spectrum antibiotics is known to eliminate the commensal microbiome as well, increasing the incidence of infections by enterobacteria and other opportunistic species. It is clear that more targeted approaches are necessary to control bacterial infections.

An element of this work concerns bacteriophage vs. bacterial interactions. These interactions are relevant to global ecology as their sheer scope creates global phenomena. Viral lysis in the oceans, for example, may directly mediate sequestration of more than 3 gigatons of carbon every year (Suttle 2007). Phages have been found in Antarctic desert soil, Pacific deep sea vents, California's hypersaline Mono Lake, and - though a decidedly less extreme environment - silty Delaware soil (Srinivasiah 2008). Phage life cycles may have significant impacts on ecosystems of every size (reviewed in Díaz-Muñoz and Koskella 2014). In the broadest sense, this area of my study concerns the most common yet least-understood protein interactions on Earth.

Work such as that presented here offers numerous opportunities for drug development and other approaches to combating bacteria. The cross-species approach I employ is

representative of that necessary to consider the secondary impact of an antimicrobial: a targeted drug should ideally be very specific to the pathogen in question as killing non-target species may cause dysbiosis. Opportunistic *Clostridium difficile* infections provide a clear example of the danger of such a phenomenon (Abt et al. 2016). A network approach offers the opportunity to not only observe weak points in a bacterial protein interaction network (and, crucially, the proteins and processes most suitable as drug targets) but also permits rapid observation of interactors several steps away in the network.

Chapter 2 - Conservation of Proteins in Bacterial Protein Complexes

Significant portions of this work have been published in the following papers:

Caufield, J.H., Abreu, M., Wimble, C., & Uetz, P. (2015). Protein complexes in Bacteria.

PLoS Computational Biology, 11(2), e1004107.

doi:10.1371/journal.pcbi.1004107.

Rajagopala, S. V., Sikorski, P., Caufield, J.H., Tovchigrechko, A., & Uetz, P. (2012).

Studying protein complexes by the yeast two-hybrid system. Methods, 58(4),

392–399. doi:10.1016/j.ymeth.2012.07.015.

Marco Abreu and Christopher Wimble assisted with initial versions of some analyses presented here and are thanked in figure legends.

2.1 Abstract

Large-scale analyses of protein complexes have recently become available for the model bacterial species *Escherichia coli* (Hu et al. 2009) and *Mycoplasma pneumoniae* (Kühner et al. 2009), yielding 443 and 116 heteromultimeric protein complexes, respectively. I have coupled the results of these mass spectrometry-characterized protein complexes with the 285 “gold standard” protein complexes identified by the EcoCyc database. A comparison with databases of gene orthology, conservation, and essentiality identified proteins conserved or lost in complexes of other species. For instance, of 285 “gold standard” protein complexes in *E. coli*, less than 10% are fully conserved among a set of 7 more distantly-related bacterial species. Expanding the

comparison to 894 distinct bacterial genomes illustrates fractional conservation and the limits of co-conservation among components of protein complexes: just 14 out of 285 model protein complexes are perfectly conserved across 95% of the genomes used, yet I predict more than 180 may be partially conserved across at least half of the genomes. No clear relationship between gene essentiality and protein complex conservation is observed, as even poorly conserved complexes contain a significant number of essential proteins. I have identified 183 complexes containing well-conserved components and uncharacterized proteins which will be interesting targets for future experimental studies. Finally, I have assembled a cross-complex protein interaction network, underscoring the surprising extent of interactions between bacterial protein complexes.

2.2 Introduction

2.2.1 The challenge of plentiful protein interaction data

Abundant genome sequencing data have revealed astounding diversity among bacterial genomes. Even species inhabiting the same environment may share only a fraction of their genes, raising the question of how these organisms have adapted to their environments in seemingly independent ways. Here, I investigate the protein complements across bacterial genomes, how proteins are combined into protein complexes across species, and whether these complexes have been conserved across diverse branches on the bacterial branch of the tree of life.

Numerous studies of protein-protein interactions have revealed the organization of proteomes into networks of interactions. Though much of this field of study focuses on individual biological pathways or protein complexes at a time, some studies have attempted to map entire interactomes of every protein-protein interaction among the members of a proteome (See **Table 1-A** for a selection of relevant studies). Few studies have included systematic surveys of protein complexes alone. Just few bacterial species, namely *E. coli* (Arifuzzaman et al. 2006; Hu et al. 2009) and *Mycoplasma pneumoniae* (Kühner et al. 2009) have been the topics of such projects. Studies of protein complexes and those of interactomes are, in theory, complementary: interactomes provide raw, generally unbiased sets of biomolecular interactions, while protein complex sets provide context for interactions (though both approaches make numerous methodological assumptions about protein relationships, of course). In practice, the limited number of data sets and the inherent differences across different species render interaction data sets difficult to interpret.

Previous research has compared the protein interaction networks of *S. cerevisiae*, *S. pombe* and *E. coli* and has found notable differences in their structure and content (Ryan et al. 2012; Dixon et al. 2009; Wuchty and Uetz 2014). Components of interaction networks also appear to vary in biological importance across species: a comparison of *S. cerevisiae* and *S. pombe* found numerous essential genes in one species were not essential in the other and vice-versa (Ryan et al. 2013). This phenomenon may be a result of methodological differences but may also reveal functional redundancy (that is,

one species may tolerate gene deletion better than another if the gene's function can be replaced by another) or species-specific adaptations (e.g., gene deletion may disrupt a crucial metabolic pathway but may have a less deleterious effect if the metabolite can be obtained from the environment). Though not specifically concerned with protein complexes, these studies have a relevant, major finding: protein interactions can be modular with respect to biological processes.

I sought to enhance the usefulness of existing protein complex sets using a multi-pronged computational strategy. I first investigated whether the complexes found in a few model organisms are sufficient to reconstruct homologous protein complexes in other species. This is a particular challenge with members of the Eubacteria as the genomes of most species are highly divergent from the few model species used here. However, *E. coli* and *Mycoplasma pneumoniae* provide two important paradigms: *E. coli* is a generalist capable of living in a variety of environmental conditions while *M. pneumoniae* is a specialized parasite with a reduced genome and a reliance on its host cell - usually a human respiratory tract cell. With ~4,300 and ~700 genes in their respective genomes, these bacterial species represent medium-sized as well as minimal genomes. Most crucially, and as noted above, both species have been subjects of protein complex surveys.

Existing studies comparing sets of interactome data have generally limited their comparisons to a few well-characterized protein-protein interaction networks, such as

comparisons of *S. cerevisiae*, *S. pombe* and *E. coli* (de Matos Simoes et al. 2013; Wiles et al. 2010; Dixon et al. 2009; Sharan et al. 2005). Methodological frameworks for predicting co-evolution on the basis of gene presence/absence (Ryan et al. 2012; Cohen et al. 2013) may also be employed to predict novel interactions in other species. Here, I initially use eight distinct bacterial species as models. Seven of these species have been subjects of gene essentiality screens and two have comprehensive, affinity purification/mass spectrometry derived protein complex surveys available. I then expand the focus to a set of 894 bacterial genomes in order to predict patterns of protein complex evolution across the entire tree of known bacterial life.

2.2.2 Extending interaction analysis across species

Few studies have investigated the evolution and diversity of protein complexes across a wide range of taxa. We may now perform such studies much more easily than in previous decades due to the prevalence of large experimental studies and a wealth of bacterial genome sequences. The number of publicly-available, sequenced bacterial genomes increased more than one hundredfold - from just two genomes to three hundred - between 1995 and 2006 and increased another hundredfold by 2015. Last year, NCBI Genbank reached a total of more than 30,000 bacterial genome sequences (Land et al. 2015), and as of September 2016 had – as mentioned above in Chapter 1 – entries for more than 70 thousand bacterial genomes at various stages of completion. Even individually, these sequences permit integrative analyses with proteomic data sets, yet we can specifically use these sequences together to evaluate the extent to

which protein complexes are likely to be conserved across microbial species.

Furthermore, we can evaluate the biological role of proteins and complexes of unknown function across many species.

I began comparing well-characterized protein complexes across species (in this case, comparing model eukaryotes) to determine the extent of their conservation (Rajagopala et al. 2012). Protein complex conservation in this context is more of a function of observed interactions than of the presence or absence of complex components, especially due to my initial choice to focus on well-conserved complexes; I operated under the assumption that one version of the complex is a good model for other species. I specifically focused on the proteasome, the protein complex responsible for proteolytic breakdown of unnecessary proteins in the cells of all eukaryote and archaea species. The pore-like structure of the proteasome appears to rely on numerous protein interactions (**Fig. 2-A-A**, see also Lasker et al. 2012). No single study in any species captures all interactions seen in any other study, though taken together, the interaction results provide a nearly complete set of interactions as per the complex structure (**Fig. 2-A-B**). This combined interaction approach suggests that protein complexes and other sets of protein interactions are best studied using multiple methods (or even, when possible, using proteins from different species). Worryingly, these results also imply that small biological differences may remain difficult to observe between species, especially when results fail to capture expected interactions.

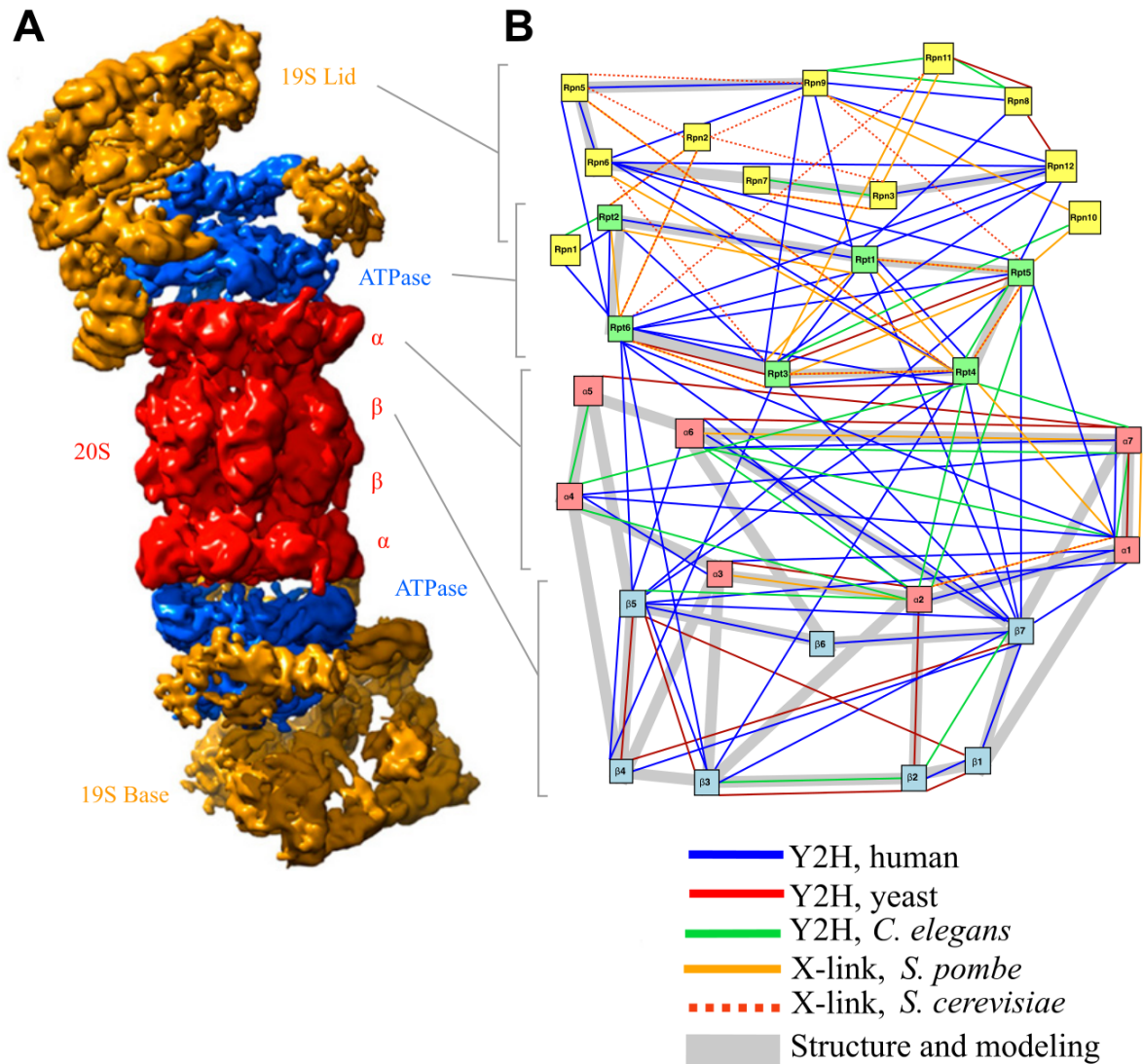


Fig. 2-A. Structure of the proteasome and interactions across five non-bacterial species. (A) Structure of the 26S proteasome from *Schizosaccharomyces pombe* as determined by cryo-EM density map (adapted from Lasker et al. 2012). The core particle is shown in red, the AAA-ATPase hexamer in blue and the Rpn subunits in gold. **(B)** Interactions among proteasome subunits as determined by Y2H and cross-linking assays (“X-link”). Four interactions between 19S proteins and beta subunit proteins are omitted for clarity ($\alpha 3$ -Rpt2, $\alpha 4$ -Rpt2, $\alpha 6$ -Rpt5, $\alpha 7$ -Rpt5). Y2H results are derived from 3 independent studies on the proteasomes of three different species (yeast, as per Cagney et al. (2001); *C. elegans*, as per Davy et al. (2001); and human, as per Chen et al. (2008)), crosslinking has been carried out in two yeast species (*Saccharomyces cerevisiae* and *Schizosaccharomyces pombe*) (Guerrero et al. (2006) and Lasker et al. (2012)). Structural and modeling results are derived from cryo-EM mapping, X-ray crystallography, and molecular modeling and used as “gold-standard” interactions, shown as grey bars (Lasker et al. (2012); Wolf and Hilt (2004)). Figure originally published in Rajagopala et al. (2012).

In order to compare genomes and protein complexes across species, I coupled the results of mass spectrometry-characterized protein complexes (Hu et al. 2009, Kühner et al. 2009) with databases of gene orthology (Powell et al. 2012) and essentiality (Luo et al. 2014) to characterize interaction conservation within protein complexes.

Furthermore, I use the perspective of genome reduction to evaluate patterns across levels of protein conservation. Comparing sets of protein complexes from divergent bacterial species (in this case, *E. coli* and *M. pneumoniae*) alleviates some of the bias inherent in using a single species as a universal model. Rather, observing which protein complexes and their components are present in two otherwise distinct species allows us to draw conclusions about how crucial these components are to bacterial life.

2.3 Experimental methods

2.3.1 Scripts

I developed a software package for the analysis of clusters of orthologous groups originally defined by version 3 of the eggNOG project (Powell et al. 2012). This software package, *spicednog*, is available online at <http://github.com/caufieldjh/spicednog>. See **Appendix I** for a full guide to *spicednog*. Given a species name or taxonomic identifier, the software performs two primary functions: it retrieves lists of genetic loci (as genes and OGs) and the number of times they are found in each genome from a given set, and determines average locus and OG conservation across the same set of species. The software includes a module for converting Uniprot protein identifiers into three

different levels of OG identifiers used by eggNOG v.3 (a feature later rendered partially redundant by updates to both Uniprot and eggNOG, though useful for batch ID conversion) and a module for specifying which genomes and taxonomic IDs from a given set contain an OG, a feature not explicitly provided by updates to eggNOG. *Spicednog* includes helper functions for determining conservation of protein complex components as well. See **Appendix I** and the Github repository mentioned above for full code and details regarding the use of *spicednog*.

Data visualization was performed using R base packages and *heatmap.2* from the *gplots* package (Warnes et al. 2015). Statistical calculations, including distance and principal component/coordinate analysis, were performed using R base packages and the *vegan* package (Oksanen et al. 2013).

2.3.2 Genome and complex data sources

The full set of protein complexes from *Escherichia coli* K-12 W3110 as defined by Hu et al. (Hu et al. 2009) was assigned membership in orthologous groups (OGs) from version 3 of the eggNOG database (Powell et al. 2012) such that each protein in a complex was assigned to a single OG. The remaining loci were referred to using their original locus identifiers (in this case, their b-codes) and were retained for all further analysis. The process was repeated for all protein complexes isolated by Kühner et al. (Kühner et al. 2009) from *Mycoplasma pneumoniae* M129 and for *E. coli* protein complexes defined by the EcoCyc database (Keseler et al. 2013). A representative set

of six other species (**Table 2-A**) for which whole-genome gene essentiality data was available was selected for in-depth analysis. This species set is referred to as the focused set. Lists of all protein-coding loci for each species were obtained using the respective full proteome sets from UniProt (see **Appendix Table III-A** for taxonomy IDs corresponding to all genomes used). Essentiality data was collected from the Database of Essential Genes (Luo et al. 2014). Protein structures were obtained from the Protein Data Bank (www.rcsb.org, Rose et al. 2013) and are referenced where used.

Table 2-A. Core set of bacterial species and strains with published essentiality screen results.

Species and Strain Name	Citation for Essentiality Screen
<i>Bacillus subtilis</i> 168	Kobayashi et al. (2003)
<i>Caulobacter crescentus</i>	Christen et al. (2011)
<i>Escherichia coli</i> MG1655	Baba et al. (2006)
<i>Helicobacter pylori</i> 26695	Salama et al. (2004)
<i>Mycoplasma genitalium</i> G37	Glass et al. (2006)
<i>Pseudomonas aeruginosa</i> UCBPP-PA14	Liberati et al. (2006)
<i>Streptococcus sanguinis</i> SK36	Xu et al. (2011)

A set of 894 genomes, referred to as the large set (**Appendix Table III-A**), was also prepared using every bacterial species present in eggNOG v.3. The full set of bacterial genomes used in this version of eggNOG includes 943 unique entries, though a subset of these genomes (49 in total) were removed as they were not present in the NCBI Taxonomy database (Federhen 2012) or were determined to differ by less than 1% in sequence. The trees shown in the figures in this chapter are cladograms intended to show the general relationship between species within context of consensus taxonomy.

2.3.3 Orthologous groups

Each locus in each genome was assigned to a single orthologous group (OG) as in eggNOG v.3 (Powell et al. 2012), such that all loci were assigned to a COG, a NOG, or a bactNOG, depending upon the most widely-conserved group assignment available. See Powell et al. 2012 for details regarding OG levels; in short, COGs are the most broadly-defined orthologous groups, based on a last universal common ancestor (LUCA) of all species in the database. COGs are defined in a similar manner to methods described by Tatusov et al. 1997 and Kristensen et al. 2010.

2.3.4 Comparative proteome and complexome analysis

The general scheme for data analysis was as follows:

1. A list of all orthologous groups (OGs) was produced for each of 894 bacterial genomes found in the large set defined above in section 2.3.2.
2. Occurrence of each OG was counted in each genome, providing the *locus count*. Averaging this count over the set of all genomes provided the *locus conservation fraction* for each OG.
3. Presence or absence of each OG was counted in each genome, such that an OG was counted only once if present in a genome. This provided the *OG count*. Averaging this count over the set of all genomes provided the *OG conservation fraction* for each OG.
4. The list from step 1 was used to map OGs to the components of three sets of protein complexes. The complexes were compared to search for cross-data set complex matches. Gene essentiality was also mapped to each OG in a species-dependent

basis. 5. A list of 8 taxonomically-divergent species was selected and used to define fractional conservation and fractional essentiality of each protein complex.

The presence of each locus was determined across the entire set of bacterial species using an automated approach. For each locus and each species, presence of a locus was defined as presence of any instance of the OG containing the locus. Each locus was assigned a fractional conservation value: e.g., a locus seen in half of all bacterial species would be assigned a conservation value of 0.5 (see **Fig. 2-B** for an additional conceptual example). This presence was averaged across all loci to generate a value for average locus conservation for each genome. This value was adjusted based on locus coverage in eggNOG (e.g., if only 70 percent of the loci in a genome mapped to eggNOG OGs, the average value was reduced by 30 percent.) An identical set of comparisons were performed for all loci with predicted paralogs (that is, loci with the same OG assignment) removed prior to comparison. Subsets of selected species were also prepared such that they included only loci with the same orthologous groups as those seen in the Hu et al., EcoCyc, or Kühner et al. protein complex sets. Genome sizes were retrieved from NCBI GenBank and KEGG GENOME (Kanehisa and Goto 2000).

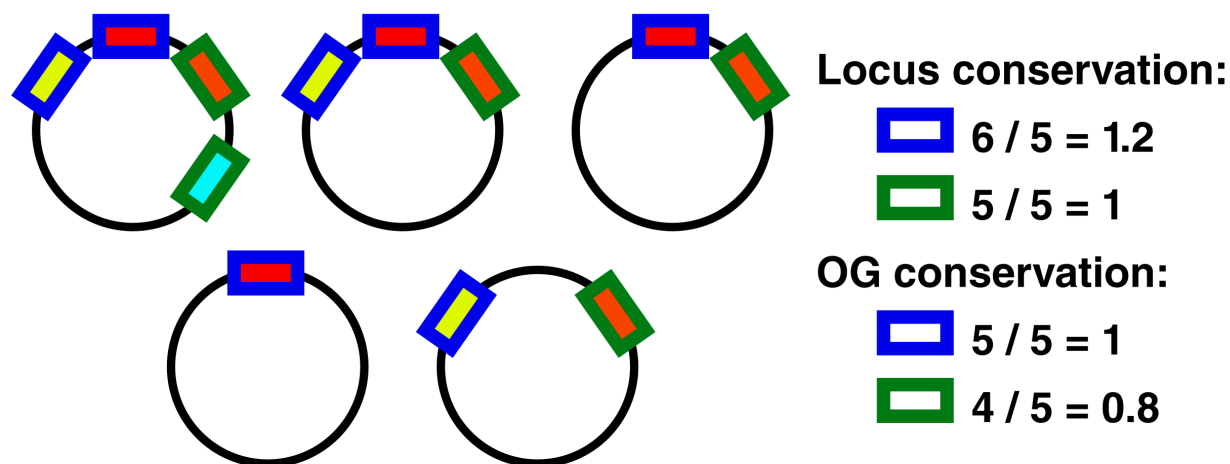


Fig. 2-B. Example of fractional conservation value assignment. Each circle denotes a unique bacterial genome. Loci are shown as filled boxes; boxes differing in outline color represent different OG assignments while fill color denotes different genes (i.e., genes with the same OG assignment in a genome are likely paralogous). Absence of a box from a genome in a location where one is present in another genome indicates absence of the locus.

OGs were used as the basis of comparison for similarity between data sets. Complex size was defined as the number of unique proteins isolated from a complex; e.g. a complex may contain 3 unique OGs but 4 distinct protein components, yielding a complex size of 4 (see **Fig. 2-C** for an example). For each complex, the presence of each OG within the complex was assayed in the full proteome sets of the seven other representative species. The resulting binary presence/absence values were combined to produce a value for the percent complex conservation. This value intentionally disregards any gene context similarity (that is, an OG may be present in two genomes even if neighboring genes differ between the genomes) and simply expresses the fraction of complex components which a specific genome may code for. When a target proteome did contain a specified complex component, the number of paralogs of the component-coding gene was determined as the number of proteins in the list mapping to the same OG. While further verification may be necessary to define any of these protein-coding genes as true paralogs, I simply used the OGs (including paralogs) as determined by eggNOG.

MdtABC-TolC multidrug efflux pump

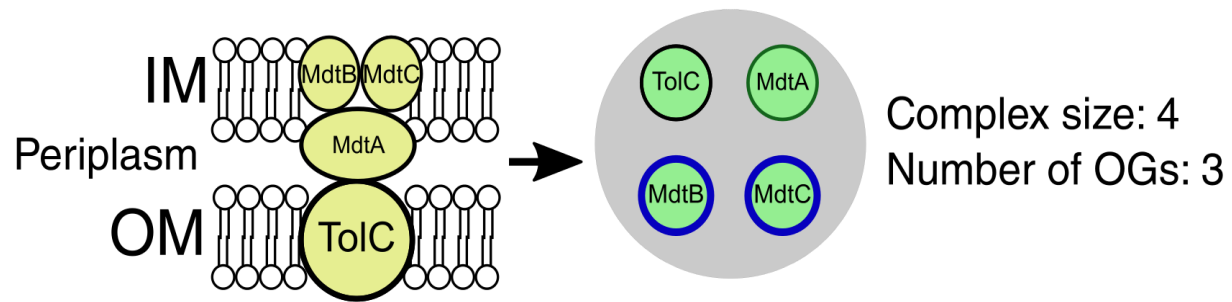


Fig. 2-C. Example of protein complex size determination. The *E. coli* multidrug efflux pump MdtABC-TolC, a transmembrane protein complex, is shown abstracted on the left (adapted from Nagakubo et al. 2002; Andersen et al. 2015). The complex is abstracted further as shown on the right, where each protein is shown as a circle. This complex involves four distinct proteins and three OGs, as indicated by the color of the outline around each circle, indicating that two of the proteins are very similar in sequence. The complex size is determined by the number of proteins rather than the number of OGs. OM, outer membrane; IM, inner membrane.

All protein complex components were also assigned binary essentiality values using published assays specific to the species listed above. These values were used to define the essentiality fraction of each potentially conserved complex, e.g. an *E. coli* complex for which 80% of the components appear to be conserved in *M. pneumoniae* but only 60% of the components may be essential in the latter species.

A broader comparison was prepared using the list of 894 species as defined above. Genome sizes for each species were retrieved from the KEGG GENOME Database (<http://www.genome.jp/kegg/genome.html>, Kanehisa and Goto 2000). For each species, the total number of OG-mapped protein-coding loci was divided by the total number of loci to produce a value for percentage mapped. Using the list of all OGs in the species, each OG was compared with all other species to determine its conservation across Eubacteria. Adjusted average locus conservation for a particular genome, $C_{AAL}(g)$, was calculated as:

$$C_{AAL}(g) = m \frac{\left(\frac{\sum C_L(g)}{L(g)} \right)}{N}$$

where C_L is the number of genomes in which the locus is present, $L(g)$ is the number of loci in the genome, N is the total number of genomes, and m is the percentage of loci mapped by eggNOG v.3. Values are adjusted using the fraction of loci actually mapped so unmapped loci lower the effective conservation.

An identical list of values, but with repeated OGs reduced to a single occurrence, was averaged to produce average OG conservation. This modification removes the effect of counting loci more than once when they share OGs, as may happen when two or more loci are paralogous. Adjusted average OG conservation for a particular genome, $C_{AAO}(g)$, was calculated as:

$$C_{AAO}(g) = m \frac{\left(\frac{\sum C_L(g)}{O(g)} \right)}{N}$$

where C_L is the number of genomes in which the locus is present, $O(g)$ is the number of unique OGs in the genome, N is the total number of genomes, and m is the percentage of loci mapped by eggNOG v.3.

The large set of bacterial genomes, as defined above in section 2.3.2, was used. Genomes were sorted by size in bases and compared to the average conservation values. For a subset of species, the Average Locus and Average OG Conservation values were calculated using only OGs found in published protein complex data sets.

Mapping of fractional complex conservation across species was performed as follows for both the focused set (8 species) and the large set. A cladogram of all species in the set was prepared using the Interactive Tree of Life (iTOL, Letunic and Bork 2011) project as per NCBI taxonomy. All protein components were mapped to eggNOG v.3

OGs and complex size was determined as defined above. Conservation fraction of each complex in each species was defined as the number of complex component OGs shared between the model (an *E. coli* complex) and the target genome over the size of the model complex. Heatmaps were prepared using the R *heatmap.2* function in the *gplots* package. Randomized models of the large set heatmaps retaining the same species order but with a randomized distribution of conservation fractions were prepared using the R function *randomizeMatrix* (in the *picante* package, Kembel et al. 2010) and the ‘richness’ null model to respect overall conservation levels.

Complex functions were assigned to each complex in the EcoCyc set manually, using a combination of EcoCyc annotations and the GO terms associated with each complex component. This was performed only for heteromer complexes (that is, complexes containing proteins of at least two different sequences rather than multimers of single proteins). These functional categories are intended to be sufficiently ambiguous to cover the wide range of potential functions of a particular complex. The functional categories are listed below in **Table 2-B**.

Table 2-B. List of general functional categories used to describe protein complex function.

Complex Functional Category	Corresponding OG Functional Categories
Cell Division	D, M
Chaperone, Protein Assembly, or Modification	O
Defense/Survival/Stress Response	V
DNA Replication, Repair, or Modification	L
Metabolism	C, G, E, F, H, I, P, Q
Motility/Chemotaxis	N

RNA Modification	A
Transcription or Transcriptional Regulation	K
Translation or Translational Regulation	J
Transport	T, U
Unknown	R, S

Complex Functional Categories are those used in this study. Corresponding OG Functional Categories are those defined by Tatusov et al. (2003) and later adapted by eggNOG (Powell et al. 2012). OG Functional Categories not applicable to bacterial biology (e.g., nuclear structure) are not included.

2.3.5 Protein complex interaction network assembly

A graph of interactions among protein complexes was constructed using a set of protein interactions specific to *E. coli*. Interactions were obtained from the IntAct database of molecular interactions and are identical to those used in Chapter 3, though filtered specifically to any strain of *E. coli*. For this reason, the interactions extend beyond any single *E. coli* interactome (e.g., Rajagopala et al. 2014). Each protein interactor was assigned a UniProt identifier and assigned to one or more protein complexes as defined by the EcoCyc set described above. Individual proteins were also assigned bacteria-level orthologous groups (bactNOGs) from eggNOG v.4 (Huerta-Cepas et al. 2015). Complexes were assigned general functional categories as described above. The interaction graph was assembled in Cytoscape (Shannon et al. 2003) 3.4.0 using only interactions between or within heteromer protein complexes. Repeated interactions and self-interactions were removed from the final graph.

2.4 Results and discussion

2.4.1 Conservation of proteins across bacterial genomes

I begin with the broadest possible view of protein conservation across a representative sample of all bacterial species (**Figure 2-D**, see also **Appendix Table III-A**). Here, the genome of each bacterial species is a point on a plot of genome size vs. average conservation of the genes in that genome, with predicted paralogs (average locus conservation, in orange) and without (average OG conservation, in blue). Average gene conservation is the average of the fractional gene conservation of all genes in the genome (e.g., a very minimal genome of 100 genes in which 50 genes appear to have orthologs in all other species – yielding a conservation value of 100% - and 50 genes fully unique to that genome and those of 9 related species – yielding a conservation value of 10 out of 894 or about 1.1% – would have an average gene conservation of about 50.5%).

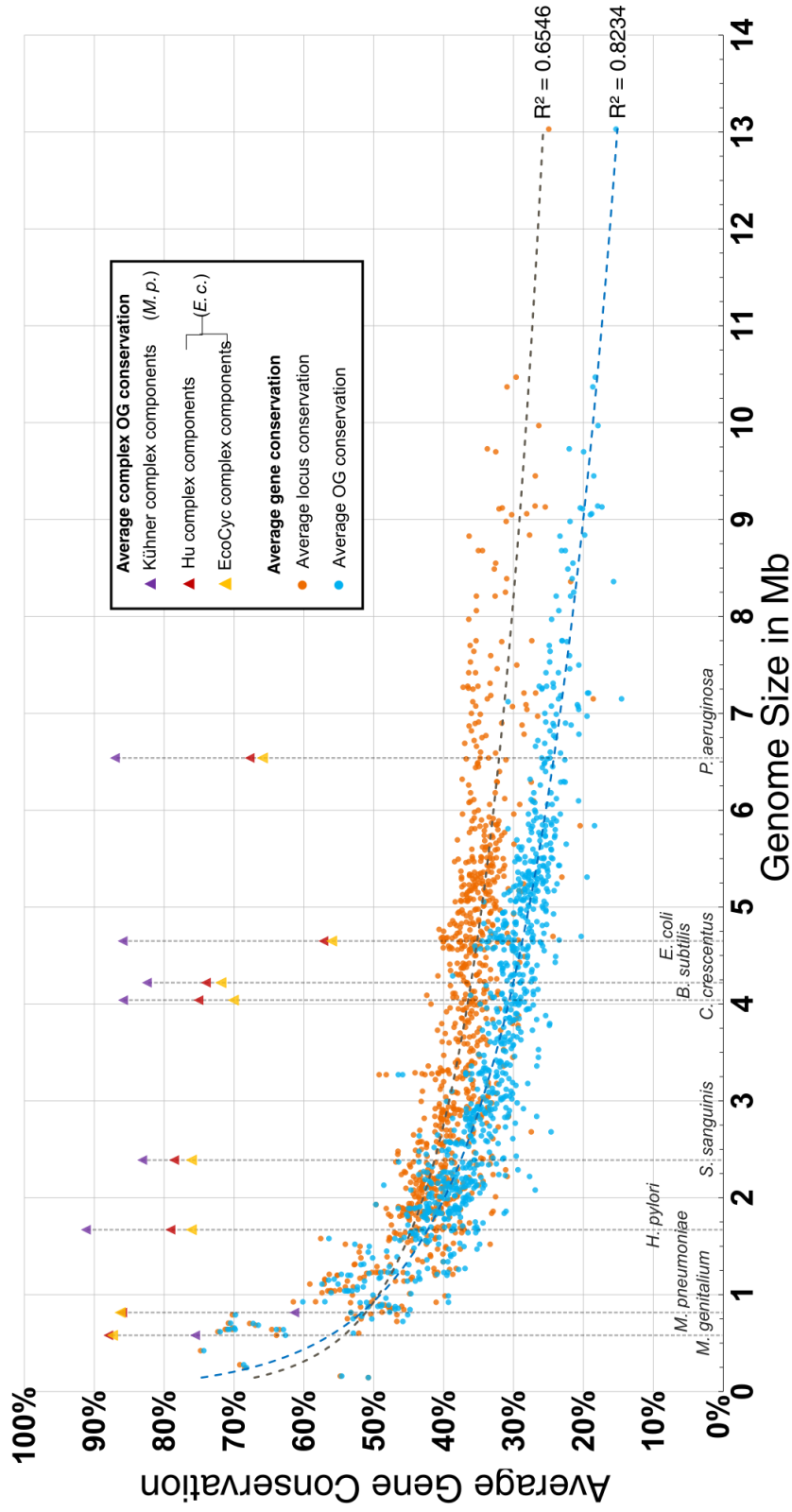


Fig. 2-D. Protein complexes are enriched for highly conserved components.

Each point indicates a single genome and the average conservation of its loci or orthologous groups (OGs) as measured by its presence across 894 bacterial genomes. Representative genomes of the 8 species focused on in this study are indicated with vertical lines and the following labels: *M. g.*, *Mycoplasma genitalium*; *M. p.*, *Mycoplasma pneumoniae*; *H. p.*, *Helicobacter pylori*; *S. s.*, *Streptococcus sanguinis*; *C. c.*, *Caulobacter crescentus*; *B. s.*, *Bacillus subtilis*; *E. c.*, *E. coli*; *P. a.*, *Pseudomonas aeruginosa*. See Methods for specific genome identities. Average gene conservation is specified as a percentage. Average gene conservation values are reduced by the fraction of their predicted protein-coding genes not present in eggNOG v.3 to account for genes without predicted orthology. To produce OG conservation instead of locus conservation, all but one locus of a set of potential paralogs (in this case, genes sharing the same OG) was removed prior to calculating averages. A logarithmic regression is fitted to both sets of values. Average OG conservation values are also shown for subsets of protein-coding genes present within protein complexes from *E. coli* (Hu et al. 2009) and *M. pneumoniae* (Kühner et al. 2009). For these two species, values are representative of members in full complexes while those for other species are predicted complexes using each of the three data sets as models.

The distribution of gene conservation values appears to follow a logarithmic regression. This trend is especially noticeable for two reasons, both of which may have been missed in a survey of a smaller range of genome sizes. First, average gene conservation of species with genomes smaller than 1 Mb can be as much as double that of species with genomes of 3 to 4 Mb or greater. Additionally, there is a gradual but consistent decrease in average gene conservation as genome size increases. The first of these observations does not appear to be impacted by including potential paralogs; this is not surprising as smaller bacterial genomes like *Mycoplasma genitalium* tend to contain fewer paralogs. In some cases, these minimal genomes may contain multifunctional predecessors or alternatives to otherwise paralogous genes, as per Mushegian and Koonin 1996; Glass et al. 2006 found that *M. genitalium* in particular has only 6% of its genes in paralogous gene families, vs. the average of 26% seen for other bacterial species. For the second observation, omitting paralogs from the analysis reduces average gene conservation values and renders them more consistent. This result suggests that genes with numerous paralogs, and hence major contributions to genome size, are also highly conserved across other bacterial species rather than usually resulting from isolated instances of rapid gene duplication in small taxonomic groups.

All average gene conservation values were proportionally adjusted to account for lack of orthology assignments. If just half of a genome's genes have corresponding orthologous

groups, for example, its average gene conservation was reduced by half. This adjustment is essentially equivalent to reducing the individual contribution in gene conservation of each gene without orthology assignment to zero. This adjustment is the best option to account for a lack of orthology assignment as a lack of OG membership generally indicates a gene sequence with little to no similarity to any other sequence. The sequence is therefore poorly-conserved by definition. In all but a few cases, orthology assignment is above 80%, suggesting that the distributions presented here are not simply the effect of differences in annotation or group assignment.

The extremes of the range of genomes presented in **Fig. 2-D** provide interesting examples of bacterial diversity with respect to gene conservation. The smallest genome shown here, that of *Hodgkinia cicadicola* Dsem, is just below 144 kb in size. Like other bacterial species with very small genomes, *H. cicadicola* is a symbiote – specifically, an α -proteobacterial species found only in cicadas (McCutcheon et al. 2009). Though likely enriched for highly-conserved genes essential to basic life functions, *H. cicadicola* is also a genetic outlier, with a much higher GC content than most sequenced symbiote genomes – more than 50% vs. 30% or lower – and a tendency toward alternative genetic codes (described by McCutcheon et al. 2009 in extensive detail). Within this same range of genome sizes is *Mycoplasma genitalium*, also a bacterial species only seen in a eukarote host, though in this case the host is *Homo sapiens* rather than cicadas. It should be noted that, of all bacterial species in this data set with genomes smaller than 1 Mb, all appear to be symbiotic or parasitic.

The other end of the bacterial genome size spectrum provides examples of species with lower overall gene conservation. The largest genome in this set is that of *Sorangium cellulosum* strain So ce56, a delta-proteobacterial isolate containing a genome of more than 13 Mb (Schneiker et al. 2007). The genome of this primarily soil-dwelling species contains numerous duplications, horizontally-transferred sequences, and complex regulatory sequences, potentially as a result of extensive environmental adaptation (Han et al. 2013). This and related species therefore provide an excellent example of bacterial genomes enriched for unique sequences.

Though average gene conservation appears to be related to genome size, gene conservation is generally more consistent across the orthologous components of protein complexes. This is the expected result: most bacterial protein complexes are expected to perform similar functions irrespective of species and many of these functions are crucial to essential processes. **Figure 2-D** displays the average OG conservation of protein complex components, using sets of complexes from *Mycoplasma pneumoniae* and *E. coli* as models, *among the components conserved in the given species*. This caveat is crucial: as discussed extensively here in subsequent sections, neither protein complexes nor their individual components are perfectly conserved across all bacterial species. The results shown here highlight the impact of genome reduction on protein complexes: out of all *M. pneumoniae* protein complex components, those conserved in other representative bacterial species are, on average, present in more than 80 percent

of other bacterial species. By comparison, *E. coli* complex components demonstrate more variable conservation across species.

2.4.2 The protein complexomes of *E. coli* and *Mycoplasma pneumoniae*

In this study, I use the literature-curated set of EcoCyc *E. coli* protein complexes and the protein complexes isolated by Hu et al. (Hu et al. 2009) as a set of experimentally-determined complexes for *E. coli* (**Figure 2-E-A**). The set of experimentally-determined *Mycoplasma pneumoniae* complexes identified by Kühner et al. (Kühner et al. 2009) is also included in the comparison as a distantly-related, minimal set. Though these datasets differ in content and approach, both *E. coli* data sets contain about 300 complexes. Most complexes in the EcoCyc set contain from 2 to 4 unique proteins while the Hu set contains a comparatively higher number of complexes (more than 30) containing 5 or more unique protein components (i.e., unique proteins mapping to different orthologous groups). Note that some of the Hu et al. complexes appear to represent subsets of full complexes (i.e., the full ribosome constitutes a single complex in EcoCyc but is represented by several complexes in Hu et al.). Also, the EcoCyc set is partially redundant (i.e., each RNA polymerase holoenzyme is represented as a different protein complex, as are the F_1 and F_o subregions of ATP synthase). The size of the complexes within the data set produced by Kühner et al appears to differ in distribution from those characterized by Hu et al (**Figure 2-E-A**). Specifically, most *M. pneumoniae* complexes with two or more unique members contain just those two unique proteins. The cross-species discrepancy may also result from methodology,

though Kühner et al. suggest it is representative of authentic biological differences between the two species. *M. pneumoniae* contains fewer unique proteins than *E. coli* does and this difference limits the number of unique proteins seen in any single complex.

The exact protein complexes defined by each data set differ. Pairwise comparison of presence or absence of proteins in each complex is improved by mapping components to orthologous groups but few complexes appear to be present in an identical form across all three data sets. **Figure 2-E-B** provides four examples of the types of complex matches seen across the data sets. For instance, the DNA polymerase III holoenzyme (EcoCyc: CPLX0-3803) contains 9 unique proteins as per EcoCyc but its closest match in the Hu set contains 7, including two proteins not found in any EcoCyc complex. The “missing” proteins from the EcoCyc complex are found in other Hu complexes. The Hsp70 chaperone complex (EcoCyc: HSP70-CPLX) provides another example: The *M. pneumoniae* complexes provide a better match for the EcoCyc complex than the Hu set does. Topoisomerase IV (EcoCyc: CPLX0-2424) has a good match in all three data sets though the representative Hu complex contains an additional protein (this addition appears to be the molybdenum cofactor biosynthesis protein MoaB, suggesting a potential role for this protein in providing alternative cofactors for the magnesium cation usually required by topoisomerase; see Sissi and Palumbo 2009). Lastly, RecBCD serves as an example of a good *E. coli*-specific match with no components present among the *M. pneumoniae* complexes.

In the aggregate, most EcoCyc complexes do not have reliable matches in the other experimental sets (**Figure 2-E-C**). Using all 285 EcoCyc complexes as a guide, their best matches in the other sets are classified as “good” if they contain at least half of the

same unique proteins (as members of orthologous groups) or “poor” if they contain a match of less than half of the EcoCyc complex’s components. No complex of a size greater than 4 unique proteins has a good match in both the Hu et al. and Kühner et al. complex sets. 28 complexes (9.8%) out of those of size 4 or less have good matches in both sets, and out of these, most matches are of complexes of size 2. The majority of the complexes in this size class (153 out of 246) contained at least one matching component in the Hu *E. coli* complexes but no match among the Kühner et al. *M. pneumoniae* complexes.

The overall lack of apparent similarity between complex sets is likely a combination of differences in experimental results and biological factors. It is likely that In some cases, a protein complex defined by EcoCyc may have been found in other sets as fragments, ensuring a poor match at best. This may be especially relevant to large complexes such as the GspC-O type II secretion complex (EcoCyc: CPLX0-3382) which has no matches in either of the two experimental complex sets. Hu et al. specifically mention they were unable to detect 469 *E. coli* proteins, about a third of which are membrane-associated. Kühner et al. also noted that membrane proteins were underrepresented in their complex set. In some cases, as with the topoisomerase discussed above, the complexes present in all three data sets may truly reflect high complex conservation across species, especially as these complexes include those involved in crucial functions (**Figure 2-E-B**).

2.4.3 Using protein complexomes to predict complexes conserved in other species

The set of *M. pneumoniae* complexes serves as a rough model for the complexes most commonly found across bacterial species. (See **Appendix Table III-I** for the full set of *M. pneumoniae* complexes and their conservation.) It is an imperfect model: out of 116 complexes, only 28 are fully conserved (that is, each of their components are present as orthologs) in the 7 other model species in this study (**Fig. 2-F**). On average, 54 *M. pneumoniae* complexes appear to share at least 2/3 of their components with all the other species and 81 complexes share at least half. As this value is an average, it does not take into account the differences between individual species in terms of complex conservation. For some complexes, such as complex 12 in this set (containing three enzymes: an aldolase, a glyceraldehyde-3-phosphate dehydrogenase, and a pyruvate kinase) all components appear to be present in all 7 other species with the exception of *H. pylori*, which does not appear to code for an orthologous kinase. The missing component in *H. pylori* may be replaced by a different protein or the entire complex may be an artifact of how broadly-conserved its apparent components are (that is, what seems like complex component conservation is simply broad conservation of individual proteins). Other complexes clearly reflect differences in species: Out of the four components of complex 33, all are conserved in *M. genitalium* but just three are conserved in the 6 other species. The protein specific to the Mycoplasma in this complex is Mpn642 (Uniprot: P75155), an uncharacterized lipoprotein.

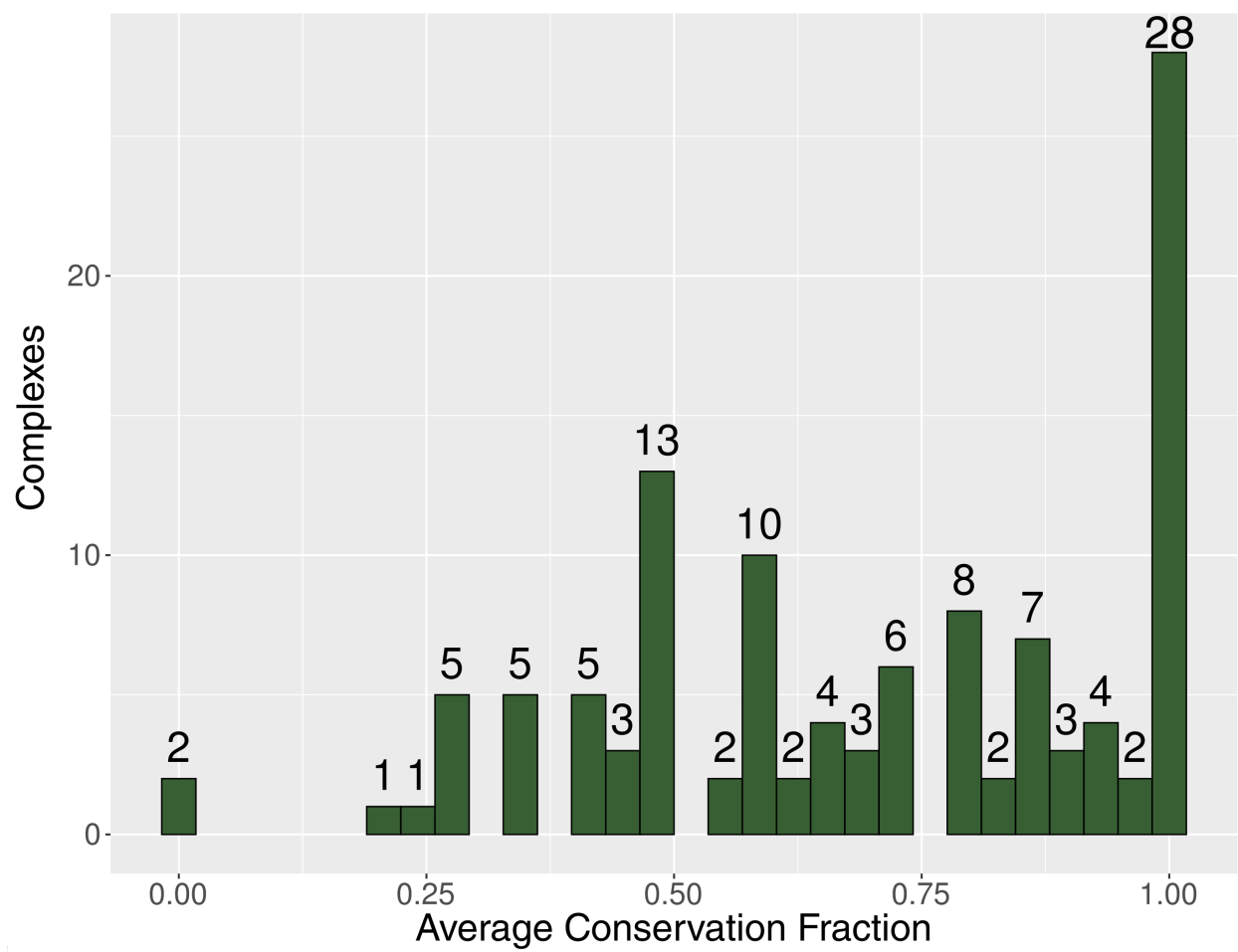


Fig. 2-F. Histogram of Kühner et al. *M. pneumoniae* complexes and average conservation fractions. Labels indicate number of complexes in each bin of average conservation fraction.

Just two complexes contain components entirely specific to *M. pneumoniae*: complex 81, composed of proteins Mpn100 and Mpn650 (Uniprot: P75592 and P75147), and complex 87, composed of proteins Mpn036 and Mpn676 (UniProt: P75078 and P75116). These two complexes and their components are uncharacterized. Due to the small number of complexes conserved in other species, the Kühner et al. *M. pneumoniae* set is not included in the majority of the subsequent analyses presented here.

The variability between the EcoCyc and Hu datasets has a direct impact on the usefulness of these complexomes as models for other bacterial species. In any case, the EcoCyc and Hu complex sets provide the most comprehensive complex set currently available for *E. coli*. The intersection of the two sets (**Figure 2-G-A**) is indeed limited: among all 1,521 unique orthologous groups seen across the two sets, just 576 OGs are shared between them. Only 132 complexes appear to be “good” matches between the sets; each set contains more complexes unique to itself than the total number of complexes shared between both sets. Using these 132 complexes as a model for those in *P. aeruginosa* shows that up to 120 of the complexes may be conserved based on orthologous components present in the *P. aeruginosa* genome. If the yet-uncharacterized *P. aeruginosa* complexome contains roughly the same number of complexes as those for *E. coli* then this prediction method misses more than half (that is, around 150) of the potential complexes unless I also use the unique complexes of each set. I used these results as evidence that the data sets should be

used as independent models rather than as an intersecting set: losing more than half of the potential model complexes simply due to inconsistencies across data sets may be too limiting for a broad cross-species comparison.

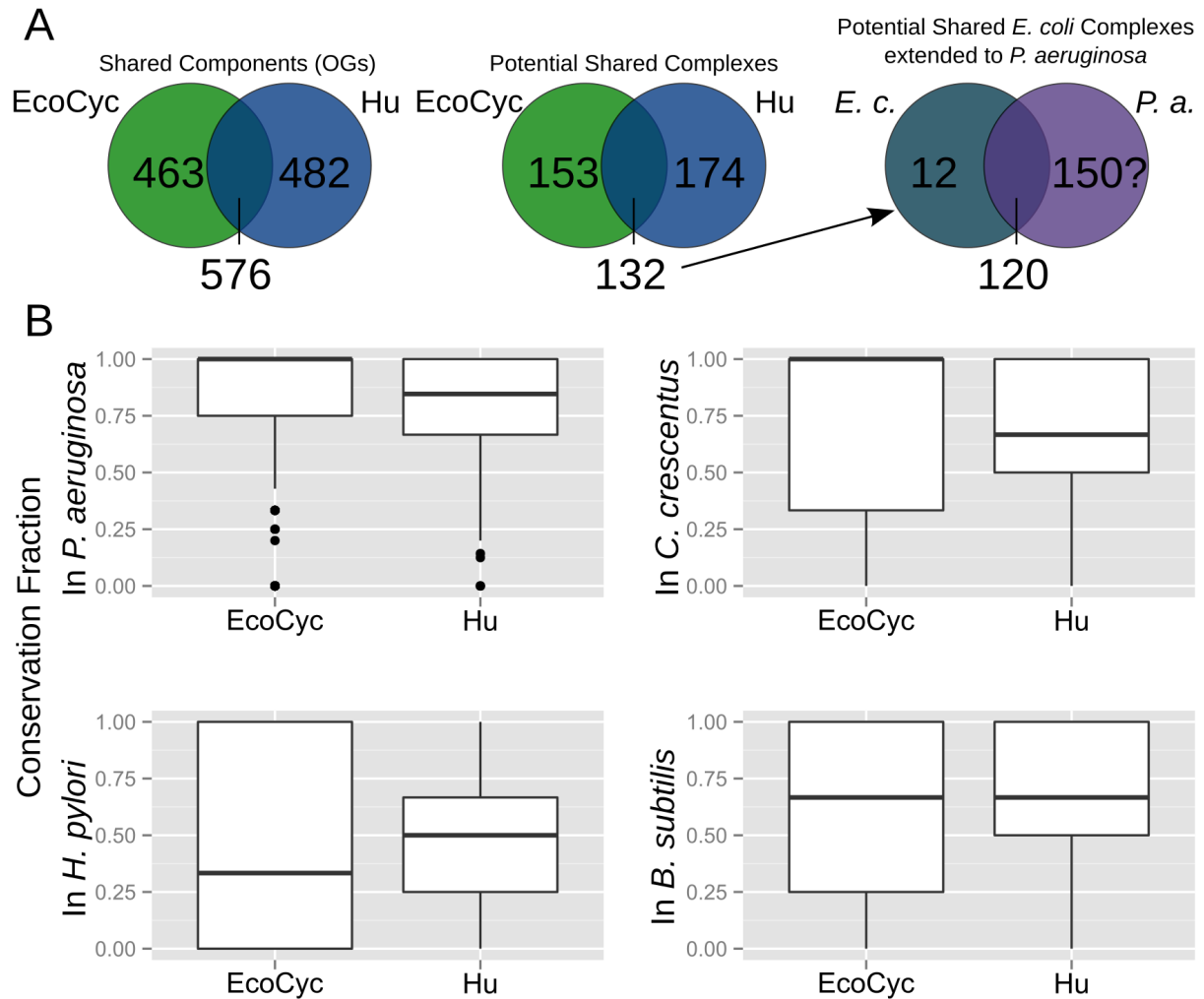


Fig. 2-G. Protein complex sets vary in conservation across bacteria.

(A) Overlap between literature-curated (EcoCyc) and experimentally-observed (Hu et al.) *E. coli* complex sets is limited. Each data set contains unique proteins, even when all are mapped to orthologous groups (far left). Each complex in one of the two *E. coli* complex sets may or may not appear to be shared in the other complex set (middle; a potentially shared complex must have at least half of its components in at least one complex in both sets). Using just the set of complexes shared between the two *E. coli* sets as a model for predicting complexomes in other species (far right; in this case, *P. aeruginosa* is used as an example) may be limiting. 12 complexes from the shared set appear to be conserved in *P. aeruginosa* but roughly an additional 150 complexes may be expected based on those seen in *E. coli*. (B) Each box plot displays the range of conservation fractions of *E. coli* protein complexes from the literature curated (EcoCyc) and experimental (Hu et al.) sets with respect to a species other than *E. coli*. The upper and lower edges of each box correspond to the first and third quartile of conservation fraction values, respectively. The upper whisker corresponds to the highest value within 1.5 times the inter-quartile range (IQR) while the lower whisker corresponds to the lowest value within the same range. Data points outside 1.5 times IQR are represented by single data points. Marco Abreu assisted with this analysis.

Fig. 2-G-B displays distributions of protein complex conservation across four bacterial species other than *E. coli*. (*M. pneumoniae* complexes are not included in this comparison.) These plots provide the median and interquartile range of protein complex conservation fractions in each species, using either EcoCyc or Hu et al. complexes as a model of the complex set. A comprehensive set of protein complexes has not been identified for any of these species as of yet. Following the results shown in **Fig. 2-D**, however, I may predict that most bacterial protein complex component sets should share at least half of their OGs with all other bacterial genomes, on average. Basic biology also plays a role here: we generally expect a subset of crucial protein complexes like polymerases to be well-conserved across all species. The set of all EcoCyc complexes appears to be highly-conserved in *P. aeruginosa* (the entire interquartile range lies between full and 75% complex conservation, showing the average EcoCyc complex is well-represented in *P. aeruginosa*) but shows a greater range of conservation across the three other species. This difference in conservation patterns likely reflects differing levels of conservation between different protein complexes, with some complexes demonstrating much higher conservation than others across evolutionary distances. For studies of *P. aeruginosa*, specifically, using *E. coli* protein complexes – and preferably literature-curated complexes – as a model the bacterial complex set may be realistic.

The Hu complexes show lower complex conservation median values than EcoCyc for all but *H. pylori* and lower variability for all but *P. aeruginosa*. Here, the median values are not as useful as the conservation ranges: the distance between the highest and lowest values includes every possibility from 0 to 100% conservation using either model of *E. coli* complexes. The two species most closely related to *E. coli* in this set – *P. aeruginosa* and *C. crescentus* – produce different median values and interquartile ranges between the sets across all protein complexes. Components of complexes in the two *E. coli* sets, used as models, are clearly conserved differently across bacterial species. A higher-resolution comparison is necessary to determine which complexes are highly conserved.

2.4.4 Protein complexes and their essentiality are poorly conserved in bacteria

Although the size distribution is different in *E. coli* and Mycoplasma, I hypothesized that homologous complexes should be very similar, both in size and composition. However, this is not true: few complexes share even half of their components across the data sets (**Fig. 2-E**). The majority of complexes shows less than 50% overlap between the two *E. coli* sets of EcoCyc and Hu et al. and between Hu et al. and the Mycoplasma complex set. This suggests that there are both technical (*E. coli* vs. *E. coli*) but also biological reasons (*E. coli* vs. Mycoplasma) for these differences.

To get a more global yet more detailed picture of protein complex conservation, we compared conservation across 8 bacterial species, including the two species for which

full protein complex sets exist. The EcoCyc complex set was used as a standard to which all other species were compared. **Fig. 2-H** provides three examples of the ways protein complexes may or may not be conserved across species. Conservation of protein complexes may be roughly grouped into three categories: well-conserved complexes, complexes with a core set of proteins conserved, and those in which no core set appears to be consistently conserved. As conservation and essentiality may be related to paralogy, I also compare the components of these complexes on the presence or absence of paralogs.

It is commonly assumed that highly conserved proteins must be important and thus should be essential in many cases. Interestingly, this is often not true (**Figure 2-H**). For example, the well-conserved succinate dehydrogenase components are essential in only 3 of the species shown. The four components of this complex (as defined by the default structure in *E. coli*) are present only in *Pseudomonas aeruginosa* and *Caulobacter crescentus*. *Helicobacter pylori* and *Bacillus subtilis* encode 3 out of 4 components and the other 3 species appear to have lost the entire complex. Similarly, the Bam outer membrane protein assembly complex (EcoCyc: CPLX0-3933) shows partial essentiality across the complex in 4 species though its components are well conserved in only 3 species. This complex has a similarly patchy pattern of conservation, with any number from zero to all 5 components conserved. In the case of *H. pylori* Bam complex, what initially seems like a lack of conservation may be the result of component replacement by functionally similar proteins (Keseler et al. 2013). By contrast, F₁ ATP synthase is conserved in all species examined. These examples show that most complexes are less well conserved than their often important functions indicate (as measured by the presence of essential proteins in these complexes).

Fig. 2-I-A displays all EcoCyc *E. coli* complexes with at least one component present in *M. pneumoniae*. In this case, fraction of essentiality (the number of protein components found to be essential out of all protein components present) is shown. **Fig. 2-I-B** displays conservation fractions of all *E. coli* complexes with at least one protein

conserved in *M. pneumoniae*, though not necessarily present in a *M. pneumoniae* complex. A complete survey of all EcoCyc complexes across these species in terms of conservation and essentiality is provided in **Figures 2-I-C** and **2-I-D**, respectively.

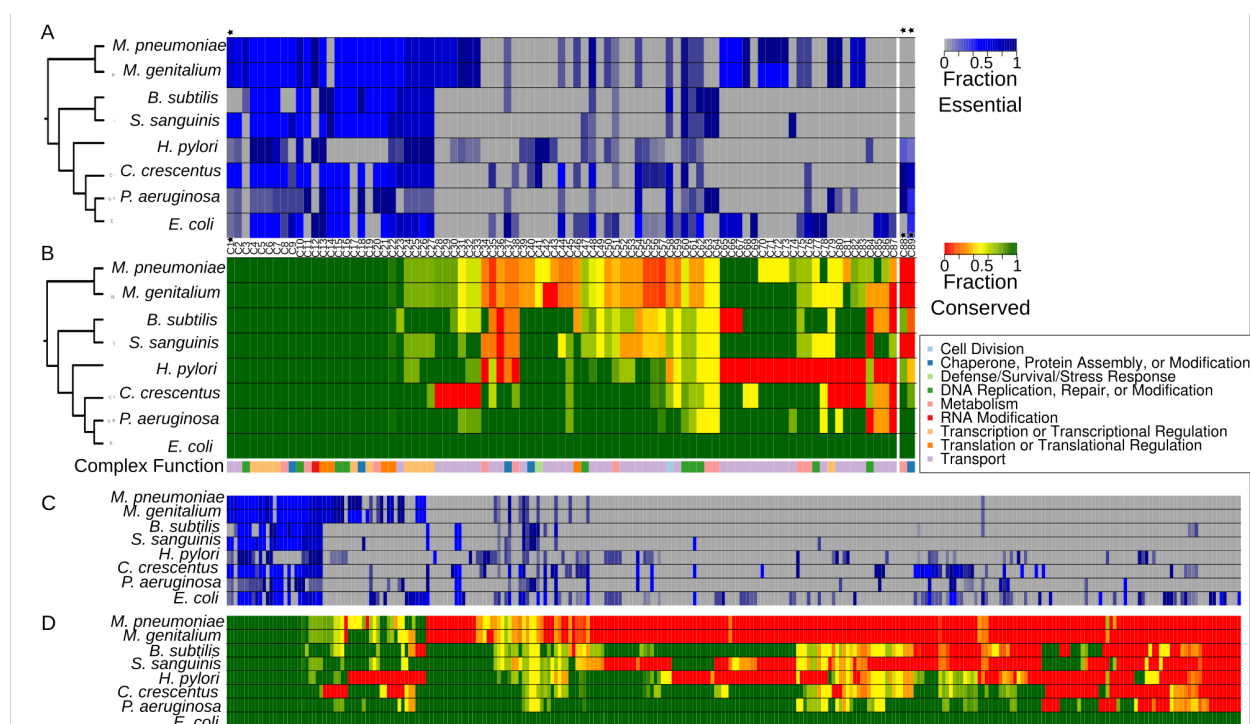


Fig. 2-I. Fractional essentiality and conservation of protein complexes across species.

(A) Each column represents one protein complex (as defined in EcoCyc for *E. coli*) and its fraction of essentiality within the species shown at left. This subset of complexes are those in which at least one component is predicted to be conserved in *M. pneumoniae*. Two example complexes not predicted to be present in *M. pneumoniae* are also shown at the far right of the complex list. See **Appendix Table III-D** for key to complexes. For species other than *E. coli*, complexes are predicted using orthologous groups (OGs). Colors indicate the fraction of essentiality: blue—conserved components are essential at the fraction specified at right, grey—no components are conserved or all conserved components are not essential. **(B)** Conservation of complexes as shown in (A). Colors indicate the fraction of conservation ranging from dark green (all proteins are present) to red (no protein is present). General functional group assignments were manually assigned based on EcoCyc annotations. Columns in panels A and B correspond to the same complexes. **(C)** As in part A, but for the full set of EcoCyc *E. coli* complexes; each column is a single complex. An extended version of this heat map is provided in **Fig. 2-J**. **(D)** As in part B, but for the full set of EcoCyc *E. coli* complexes; each column is a single complex. The order of complexes is identical to that in (C). An extended version of this heat map is provided in **Fig. 2-K**. Columns in panels C and D correspond to the same complexes.

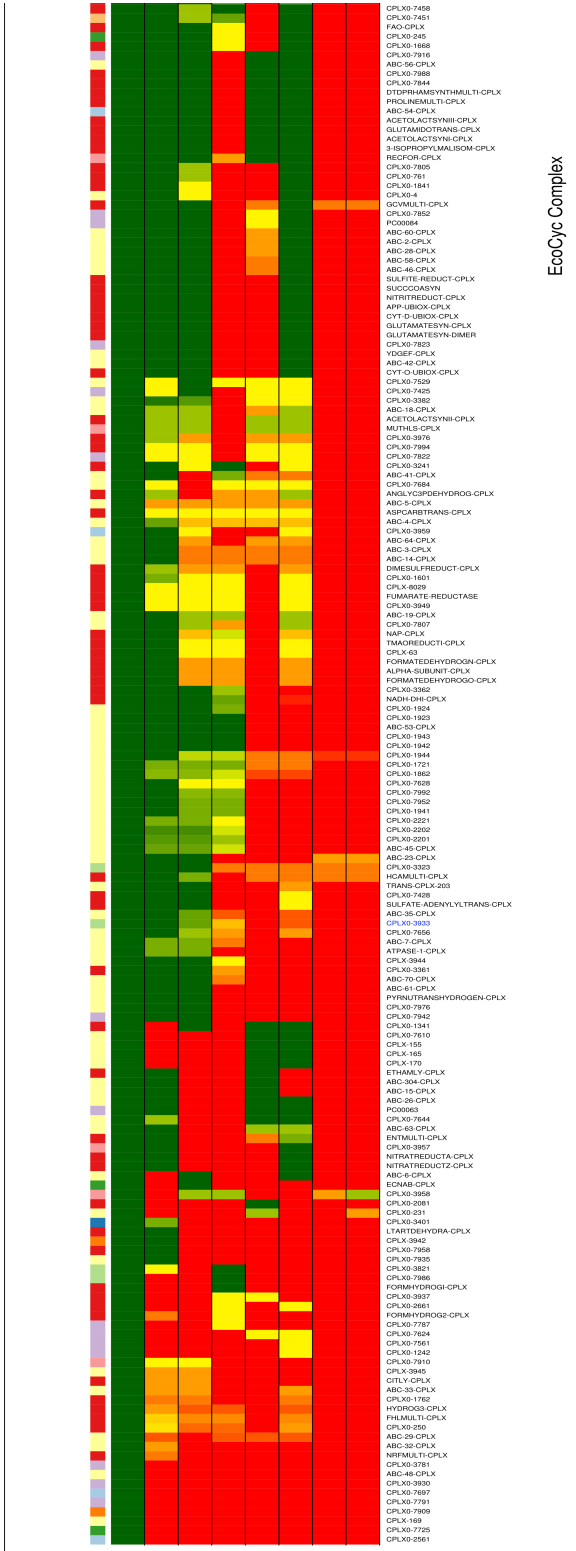
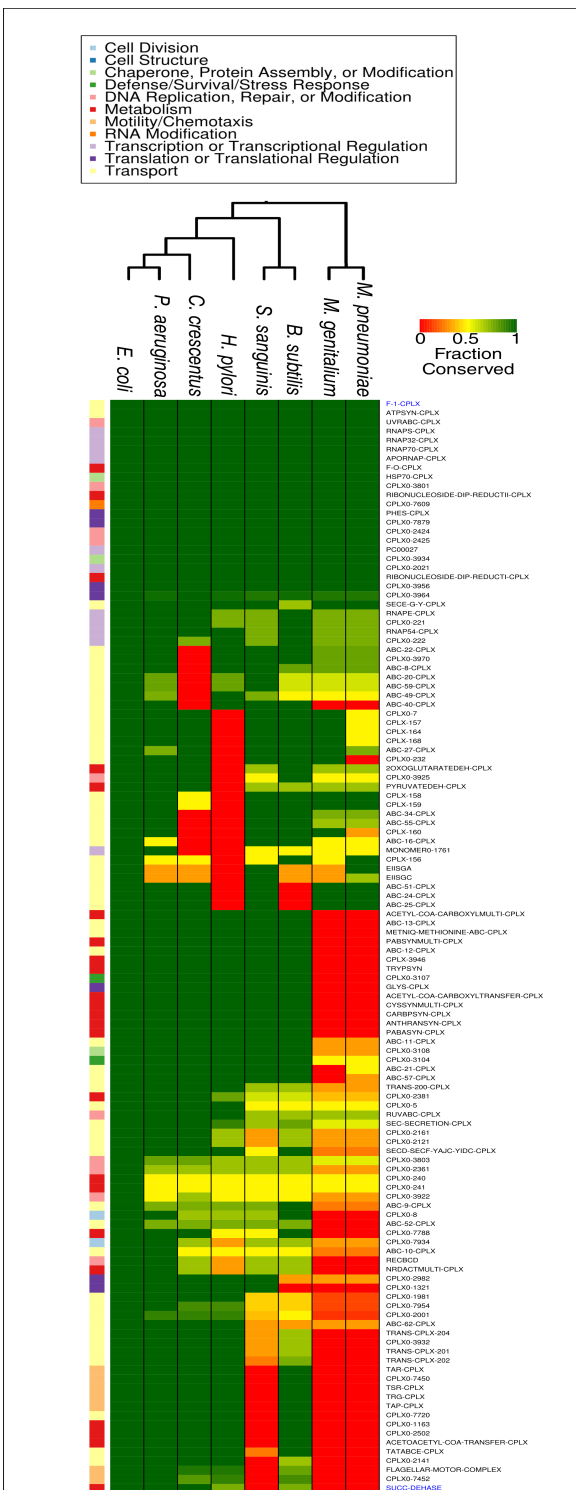


Fig. 2-J. All EcoCyc complexes and their fractional conservation in selected bacterial species. An extended version of Figure 2-I-D. Names in blue indicate example complexes shown in Figure 2-I.

Fig 2-K. All EcoCyc complexes and their fractional essentiality in selected bacterial species. An extended version of Figure 2-I-C. Names in blue indicate example complexes shown in Figure 2-I.

Conservation fraction was established as the fraction of unique proteins in a defined complex present in the target species. Notably, proteins of only 21 complete EcoCyc complexes are fully conserved across all 8 species, or just 15 complexes when subunits and alternate forms (e.g., RNA polymerase with different sigma factors) are removed. An additional 19 complexes are fully conserved across all species but the two *Mycoplasma* species, suggesting that the mycoplasmal ancestry eliminated these complexes entirely. The remaining complexes vary extensively in their degree and extent of conservation. A number of complexes are well conserved across *E. coli*, *P. aeruginosa*, *C. crescentus*, *H. pylori*, and *B. subtilis* but not *S. sanguinis* or the *Mycoplasma* (e.g. succinate dehydrogenase, EcoCyc: SUCC-DEHASE). Overall, of the 176 EcoCyc complexes of 3 or more unique proteins, 128 appear to have lost at least one unique protein component in one or more species. This demonstrates that protein complexes are far more flexible in evolutionary terms than previously assumed.

E. coli complexes serve as a “gold standard” for protein complexes across bacteria only in cases where most or all of the components of a complex are broadly conserved. This property is true of just a small fraction of complexes. **Figure 2-E-D** displays conservation fractions for all 285 *E. coli* complexes in the EcoCyc set, clustered by similarity of their conservation patterns across the 7 other species used in this study. Just 21 complexes appear to be fully conserved (that is, orthologs of each of their components are present) in all other species. This is a broad taxonomic range, so a

more relaxed cutoff may be appropriate to predict a complex is conserved; even so, only 28 complexes contain at least 2/3 of the *E. coli* components across all species. Lowering the cutoff to conservation of at least half of the *E. coli* components in each complex still yields only 34 complexes. The lack of broad conservation is not, however, a matter of full complex presence or absence across species. Rather, many complex components appear to be conserved independently from other members of their complex. Similarly patchy conservation can be seen for essentiality (**Fig. 2-K**), as the most broadly well-conserved complexes (far left) generally retain essentiality across species but less consistently-conserved complexes do not, though they may retain essentiality while appearing to lose complex components.

Protein complex function varies in a similar way as conservation (**Fig. 2-J**). As expected, many of the most highly conserved complexes are directly involved in DNA replication, transcription, or translation. Many protein complexes of varying conservation fractions are transport complexes – as bacterial membrane structures vary across species, some degree of transporter component evolution is also expected. At least six distinct complexes involved in DNA modification or repair demonstrate less than perfect conservation.

2.4.5 The *E. coli* protein complexome as a model for other species

E. coli is frequently used as a model organism for bacteria in general. Using the literature-curated set of protein complexes from EcoCyc, I sought to determine how well

this protein complexome serves as a model for complexes in other bacterial species. A comparison of the fractional conservation of each EcoCyc complex across 894 different bacterial genomes was the result (**Fig. 2-L**). The genomes in this comparison were arranged as per NCBI taxonomy definitions, revealing patterns in complex conservation closely corresponding to numerous taxonomic boundaries. Hierarchical clustering of each *E. coli* model complex (specifically, UPGMA) on the basis of its fractional conservation across all other species reveals groups of complexes with similar patterns of predicted conservation.

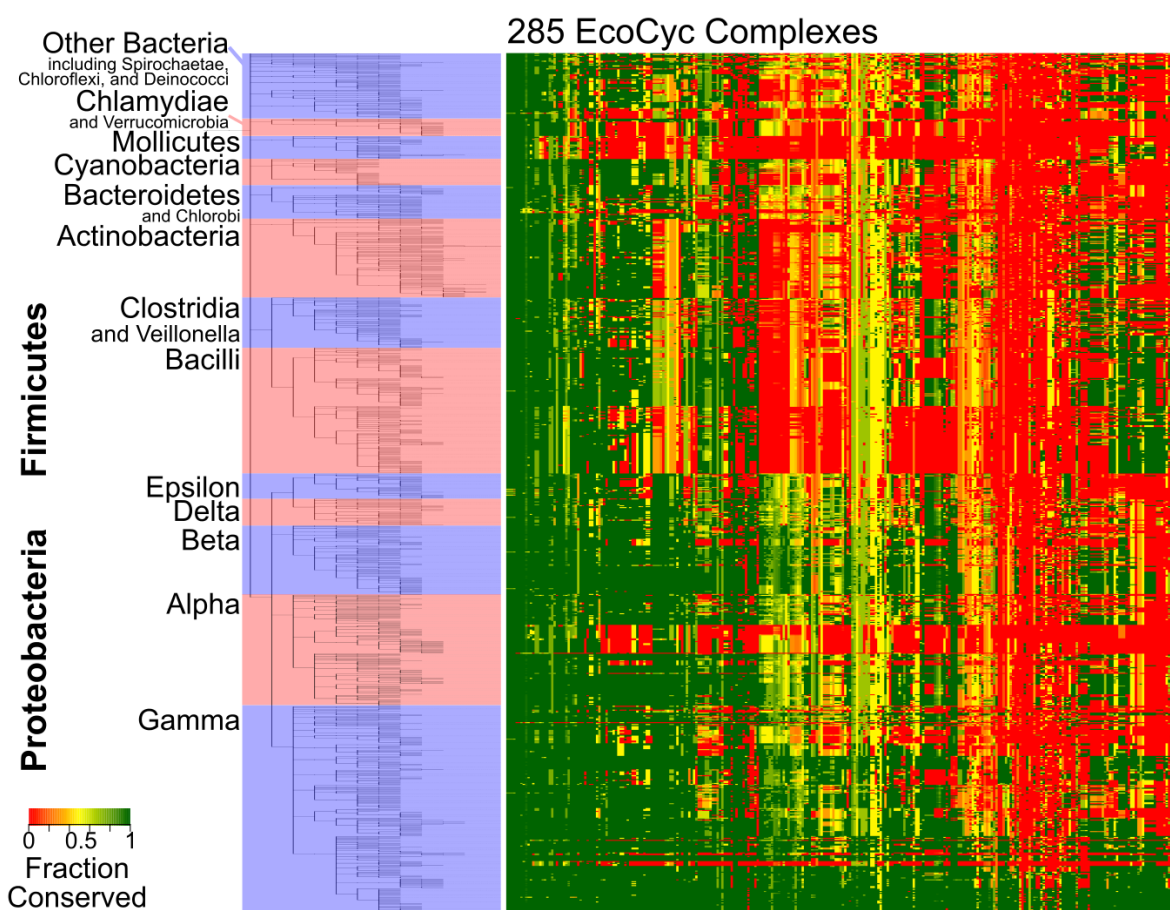


Fig 2-L. *E. coli* complex conservation across Bacteria corresponds to taxonomic boundaries. The heat map displays fractional conservation of EcoCyc protein complexes as in Fig. 2-E, though in this case across all 285 complexes in the set and across 894 different bacterial genomes as indicated on the tree at left. See Materials and Methods for taxonomic details. Tree follows NCBI taxonomy. Complexes (columns) have been clustered on the basis of the distance between their average fractional conservations (using average linkage).

The species with the most overall conservation of the *E. coli* complexes are, unsurprisingly, those most closely related to *E. coli*. Roughly a third of the complex set is conserved across all species with the minimal Rickettsia and Mycoplasma genomes, among others, serving as notable exceptions. This is a crucial distinction between the data shown in **Fig. 2-L** and that in **Fig. 2-I**: the small set of representative species fails to highlight the full range of bacterial genome diversity and therefore does not capture complex conservation in numerous species (e.g., all Clostridia and Actinobacteria). The middle third shows the most difference in conservation between the Proteobacteria and all other species. The Lactobacillales show the most difference in conservation among these complexes, to the degree that they resemble Cyanobacteria more closely among this subset. The last third (far right of **Fig. 2-L**) of the complexes demonstrate the most variable conservation across all species.

Many of these complexes are missing or partially conserved among the Proteobacteria yet are fully present in many Firmicutes species and even in extremophiles like *Thermus* or *Thermotoga* species. Overall, out of 285 EcoCyc complexes, 12 (~4%) have at least one component present in all 894 bacterial genomes in the set. None are perfectly conserved across all genomes but 14 complexes appear to be conserved across at least 95% of the genomes. This level of conservation is consistent with the traditional view of some complexes (e.g., ribosomes) as static in composition and function. If potential complex conservation is generously defined as conservation of at least half of the complex components, 3 EcoCyc complexes are potentially conserved

across all 894 genomes, 25 are potentially conserved across 95% of the genomes, and 186 are potentially conserved across at least half of the genomes. Variance across the full set of complex conservation fractions is 0.189. Because conservation of these complexes follows the existing taxonomy well, some generally well-conserved complexes like RNA polymerases may be missing from entire genera.

The experimentally-determined protein complexes identified by Hu et al. were also used as a model of the *E. coli* complexome (**Fig. 2-M**). Few members of the Hu et al. set appear to have clear matches in the EcoCyc set (see **Table 2-C**), potentially demonstrating the variability inherent to experimental results, but also suggesting the Hu et al. set may contain complexes not included in EcoCyc. Most complexes appear to have partial conservation across nearly all species using this model. Distinctions are still seen among the minimal genomes of the Rickettsiales as well as the Mycoplasma and the genomes of related species. Out of 310 Hu et al. complexes, 16 (~5%) have at least one component present in all 894 bacterial genomes in the set. As with the EcoCyc complexes, none are perfectly conserved across all genomes but a single complex (complex 271) appears to be conserved across at least 95% of the genomes. Using the same 50% cutoff for potential complex conservation as used above, no Hu complexes appear to be conserved in all 894 genomes, 10 are potentially conserved across 95% of the genomes, and 182 are potentially conserved across at least half of the genomes. Though these Hu et al. complex values appear similar to those for the Ecocyc complexes, variance across the full set of Hu complex conservation fractions is 0.097,

indicating less variability among the values than that seen for the EcoCyc complexes.

This lesser variance can also be seen in the surprising consistency across taxonomic lines (**Fig. 2-M**).

Table 2-C. Hu et al. (2009) *E. coli* protein complexes and the best matches among EcoCyc *E. coli* protein complexes.

Hu et al. complex	Best match complex in EcoCyc	Coverage
2	RUVABC-CPLX	0.67
8	CPLX0-1923	0.67
10	RNAP54-CPLX, RNAPS-CPLX, RNAP32-CPLX, RNAP70-CPLX, APORNAP-CPLX	1
16	CPLX0-240, CPLX0-241	1
19	CPLX0-7852, PC00084	1
31	CPLX0-7986	1
38	3-ISOPROPYLMALISOM-CPLX	1
42	CPLX0-7910	1
43	NITRATREDUCTA-CPLXN, NITRATREDUCTZ-CPLX	0.67
44	ALPHA-SUBUNIT-CPLX,DIMESULFREDUCT-CPLX, FORMATEDEHYDROGO-CPLX	0.67
46	SUCCCOASYN	1
47	CPLX0-7935	0.8
48	CPLX0-3958	1
50	2OXOGLUTARATEDEH-CPLX	0.67
56	ALPHA-SUBUNIT-CPLX,DIMESULFREDUCT-CPLX, FORMATEDEHYDROGO-CPLX	0.67
58	CPLX0-3801	0.67
59	CPLX0-2121, CPLX0-2161	0.67
71	ACETYL-COA-CARBOXYLTRANSFER-CPLX	1
78	SULFITE-REDUCT-CPLX	1
92	CPLX0-3361	1
98	CPLX0-1668, FAO-CPLX	1
99	FORMHYDROGI-CPLX	0.67
122	CPLX0-7942	1
154	NRDACTMULTI-CPLX	0.67
191	ABC-12-CPLX	0.67
208	CPLX0-2424, CPLX0-2425	1
231	CPLX0-2502	1
241	RECBCD	1
289	GLUTAMATESYN-DIMER, GLUTAMATESYN-CPLX	1
299	ABC-6-CPLX	1
309	CPLX0-2982	0.67

Coverage is the fraction of components of the Hu et al. complex present in the EcoCyc complex. All matches shown here are those with a coverage fraction of at least 2/3. Complexes may have more than one match if the matching complexes share components (e.g., with RNA polymerase forms such as RNAPE-CPLX).

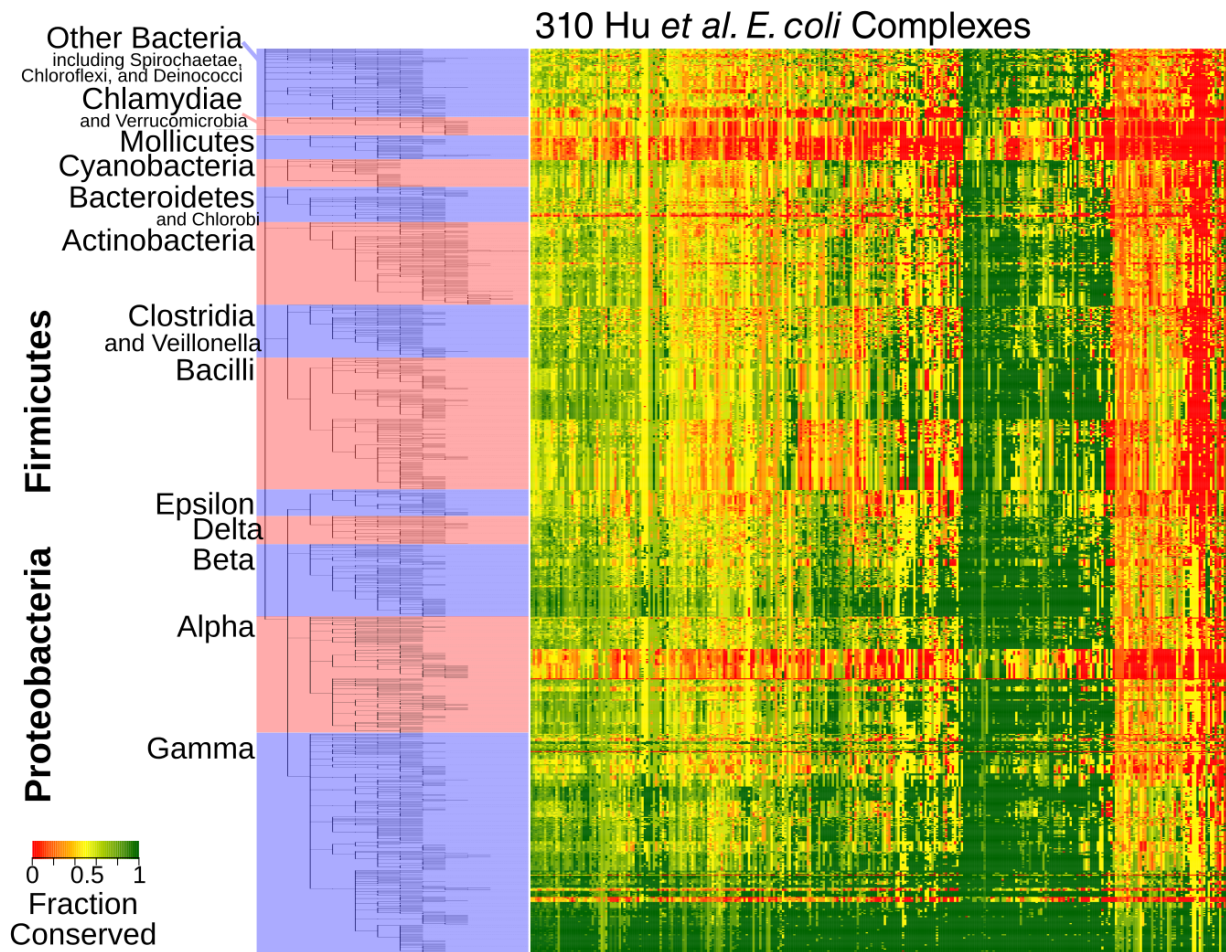


Fig. 2-M. *E. coli* experimentally-observed complex conservation across bacteria corresponds to taxonomic boundaries. Using the Hu *et al.* set of protein complexes as a model, each column is a single complex from the set and each row is a distinct bacterial genome. 310 complexes and 894 genomes are shown in total. See Methods for taxonomic details. Tree follows NCBI taxonomy. Complexes (columns) have been clustered on the basis of the distance between their average fractional conservations (using average linkage).

Both the literature-curated EcoCyc model and the Hu et al.-based experimental model were evaluated by comparison to a randomized version of their respective components. For the literature-curated model, Pearson correlation was 0.185, while for the experimental model, Pearson correlation was 0.293. The higher correlation value for the experimental model indicates it is closer to a random distribution of complex correlation fractions across the species set. I do not expect complexes to be conserved in a random pattern so this may indicate the Hu et al. complex set is less useful than the EcoCyc complex set for prediction across this wide range of genomes.

2.4.6 Essentiality of proteins in complexes and the impact of paralogy

Mycoplasma species have highly reduced genomes and it is generally assumed that they have retained mostly essential proteins. In fact, the fraction of conserved essential proteins is much higher when comparing *Mycoplasma pneumoniae* to *E. coli* than vice-versa (**Fig. 2-N**). In these comparisons, all complex components are searched for in full genomes and essentiality is assigned based on the target species. Among the full set of Hu et al. *E. coli* complexes, complexes have an average conservation fraction of 0.198 ± 0.230 and an average essentiality fraction of 0.122 ± 0.196 in *M. pneumoniae*. High variability in conservation among complexes is expected as complex components, like single proteins, are subject to a broad variety of evolutionary pressures. Among the 53% of complexes with at least one component present in *M. pneumoniae*, the average fractions increase to 0.375 ± 0.184 and 0.231 ± 0.218 , respectively. Among the full set of Kühner et al. *M. pneumoniae* complexes, complexes have an average conservation

fraction of 0.716 ± 0.292 and an average essentiality fraction of 0.32 ± 0.332 in *E. coli*.

Among the 95% of complexes with at least one component present in *E. coli*, the average fractions increase to 0.755 ± 0.245 and 0.337 ± 0.332 , respectively. Overall, Mycoplasma protein complex components are more likely to be present and essential in *E. coli* than *E. coli* protein complex components are in Mycoplasma.

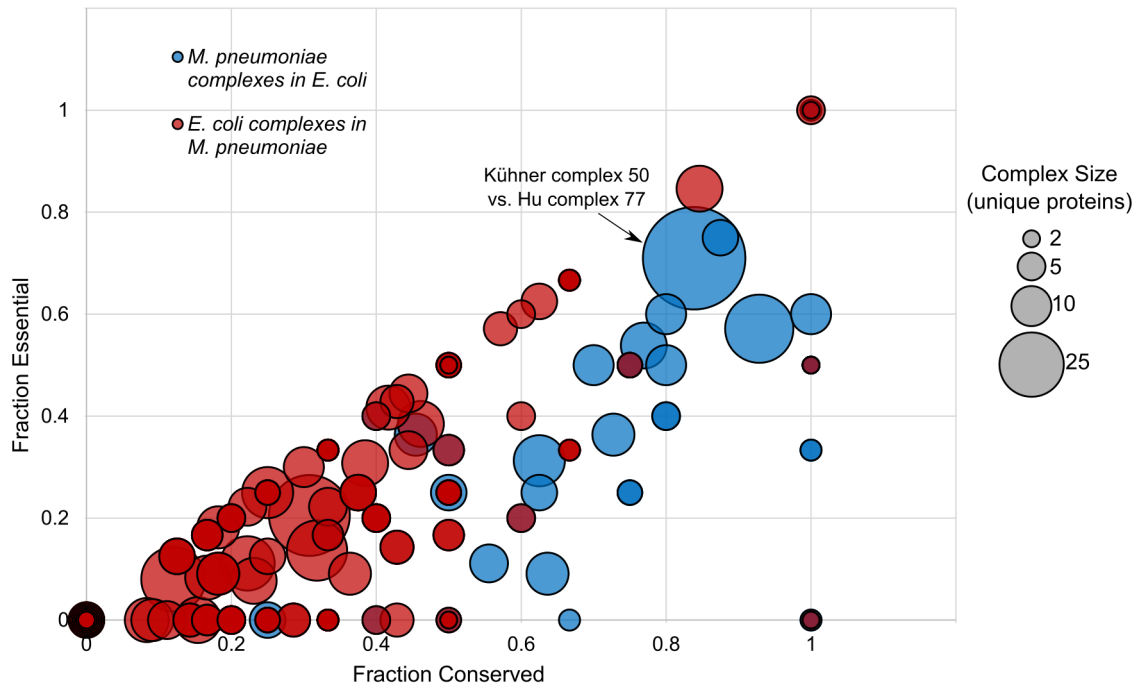


Fig. 2-N. Conserved complex components are enriched for essential proteins. This correlation is more pronounced in *Mycoplasma pneumoniae* (blue). Protein complexes of *E. coli* (Hu et al. 2009) are compared to complexes of *M. pneumoniae* (Kühner et al. 2009) and vice versa. Fraction of conservation and fraction of essentiality are calculated as described in Methods. Each node represents a single protein complex with relative size corresponding to the size of the complex in number of components. Kühner complex 50 and its corresponding Hu complex 77 are indicated as an example complex match.

One possible explanation for the lower fraction of conserved essential proteins in *E. coli* is the presence of paralogs that renders duplicate genes non-essential, given the presence of an additional copy with a redundant function. I performed comparisons of the fraction of conservation of each complex and its sum of paralogy (that is, the total number of all copies of all genes coding for the complex components in the target species, as determined by shared OG membership). As the number of paralogs for each gene is broadly defined, these numbers are considered *maximum possible values* rather than specific counts of known paralogous regions.

There is an inverse trend between *E. coli* complexes vs. *M. pneumoniae* (**Fig. 2-O-A**) and vice versa (**Fig. 2-O-B**): the more paralogs they have in *E. coli* the less conserved these proteins were in *M. pneumoniae* and vice versa. More specifically, *E. coli* complexes with a conservation fraction greater than 0.6 in *M. pneumoniae* all had total paralogy sums lower than 40 though more poorly-conserved complexes had paralogy sums between 2 and about 100. *M. pneumoniae* complexes with a conservation fraction greater than 0.6 in *E. coli* had a range of sums of paralogy between 2 and nearly 80. The more poorly-conserved complexes all had paralogy sums of 60 or less. Pearson anti-correlation for *E. coli* complexes vs. *M. pneumoniae* (**Fig. 2-O-A**) was -0.04 and Pearson correlation for *M. pneumoniae* complexes as a model for *E. coli* (**Fig. 2-O-B**) was 0.05, indicating limited to no overall correlation in either full comparison. As is the case with conservation of complexes across all species (**Fig. 2-L**), correlation is likely case-specific. The simplest explanation for this observation may be that complexes with

extensive paralogy (that is, their components have sequences with similarity to other proteins encoded elsewhere in the genome) represent evolutionary flexibility. A complex may be more tolerant of the loss of an individual component if a similar, redundant protein may take its place. Such redundancy does not appear to be present in the *M. pneumoniae* genome.

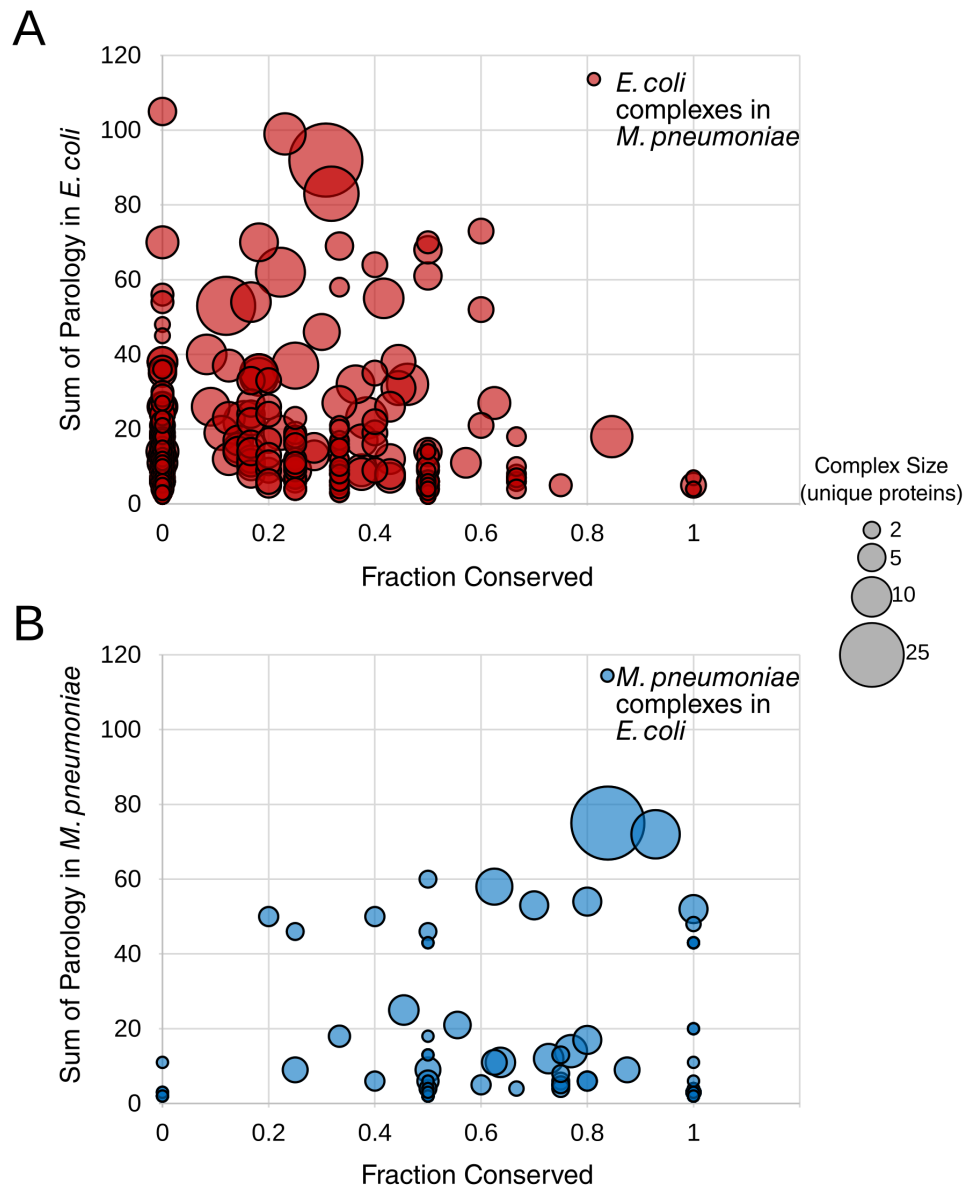


Fig. 2-O. Cross-species conservation of experimentally-observed protein complexes and the sums of the counts of potential paralogs of their components. (A) Proteins in *E. coli* complexes (Hu et al. 2009) tend to have more paralogs if the complexes are less conserved. **(B)** By contrast, in *M. pneumoniae* complexes (Kühner et al. 2009) more conserved complexes tend to have more paralogous proteins. Fraction of conservation and sum of paralogy are calculated as described in Methods. Each node represents a single protein complex with relative size corresponding to the size of the complex in number of components. *E. coli* complexes as defined by Hu et al. were compared to the full *M. pneumoniae* proteome while *M. pneumoniae* complexes were compared to the full *E. coli* proteome; all cross-species comparison are done using predicted orthologs as described in Methods.

The fraction of essential components in protein complexes is non-random and may be greater than expected, depending upon the complexes compared (Fig. 2-L). When compared to random assortment (that is, a condition with no enrichment), Hu et al. *E. coli* complexes have more essential proteins than expected (**Fig. 2-P-A**). A Spearman anti-correlation of -0.25 was found for this set. *E. coli* complexes from EcoCyc (**Fig. 2-P-B**) demonstrate similar trends, with a Spearman anti-correlation of -0.22. *M. pneumoniae* complexes from Kühner et al. (**Fig. 2-P-C**) show a trend of declining essentiality compared to randomized essentiality fractions of 0.6–0.8. A Spearman anti-correlation of -0.03 was found for this *M. pneumoniae* complex set. Both *E. coli* anti-correlations show a weak relationship.

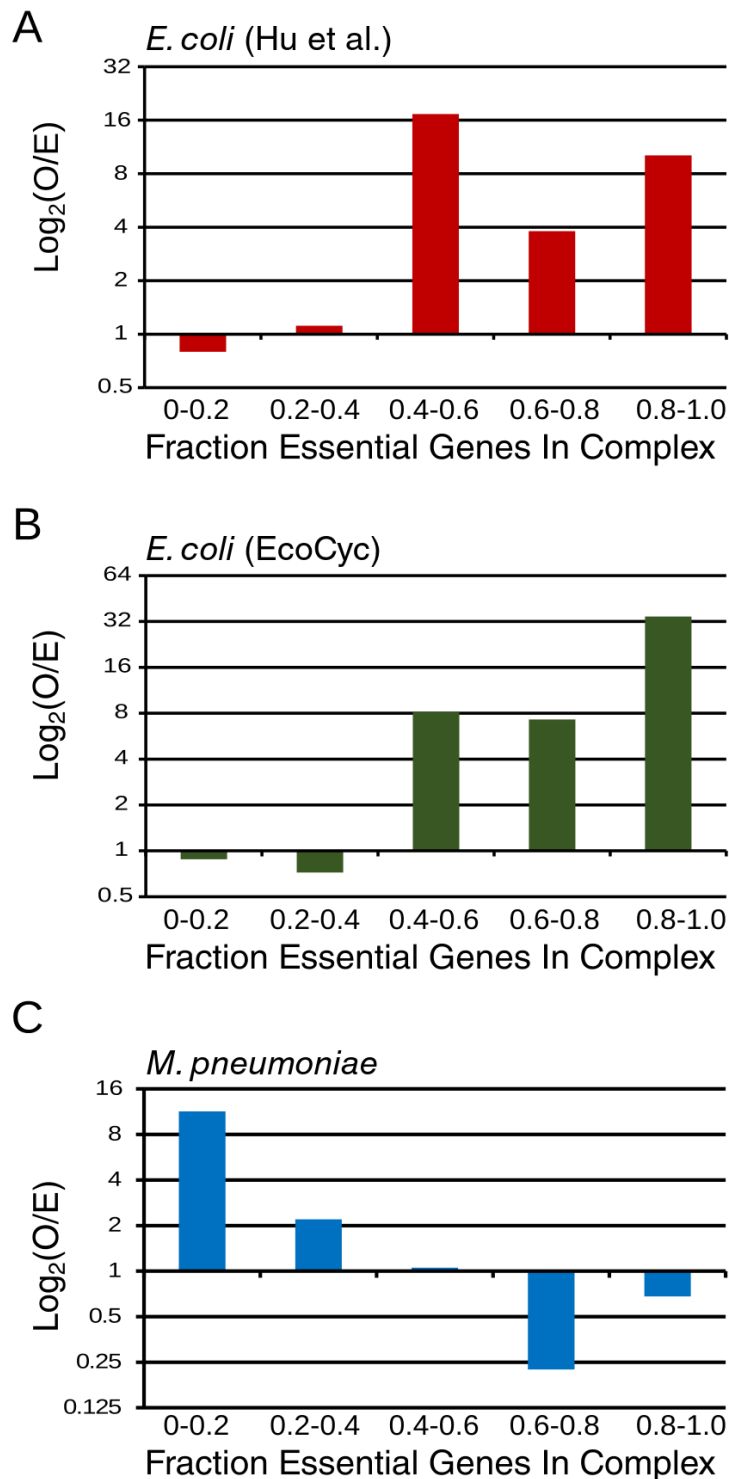


Fig. 2-P. Essentiality of proteins in complexes. Distribution of essential genes among those from **(A)** *E. coli* (Hu et al. 2009) and **(B)** *E. coli* (EcoCyc), and **(C)** *M. pneumoniae*. The fraction of essential genes within protein complexes was determined for each complex set. In *E. coli*, essential protein complexes are enriched for essential proteins. By contrast, complexes with non-essential proteins are over-represented in the genome-reduced *Mycoplasma pneumoniae*. Each distribution is expressed as binned \log_2 ratios of observed over expected frequency. Values indicate observed frequency above or below random results (= 1), respectively. Marco Abreu assisted with these analyses.

2.4.7 Proteins of unknown function

Protein complexes are attractive targets for functional analysis, given that proteins are embedded in a functional context. This is especially true for proteins of unknown function that are part of a complex (**Fig. 2-Q-A, B**). Here, conservation is defined as greater than 0.5 conservation fraction and essential complexes are those with at least one essential component in the target species. Among the highly conserved components, many are essential in 4 or more of the 8 species. Using more than one species reduces the effect of noise and inconsistency across essentiality screens. Starting with 39 EcoCyc-defined complexes containing unknown proteins, at least 15 appear to be conserved in five other species shown here other than the Mycoplasma representatives.

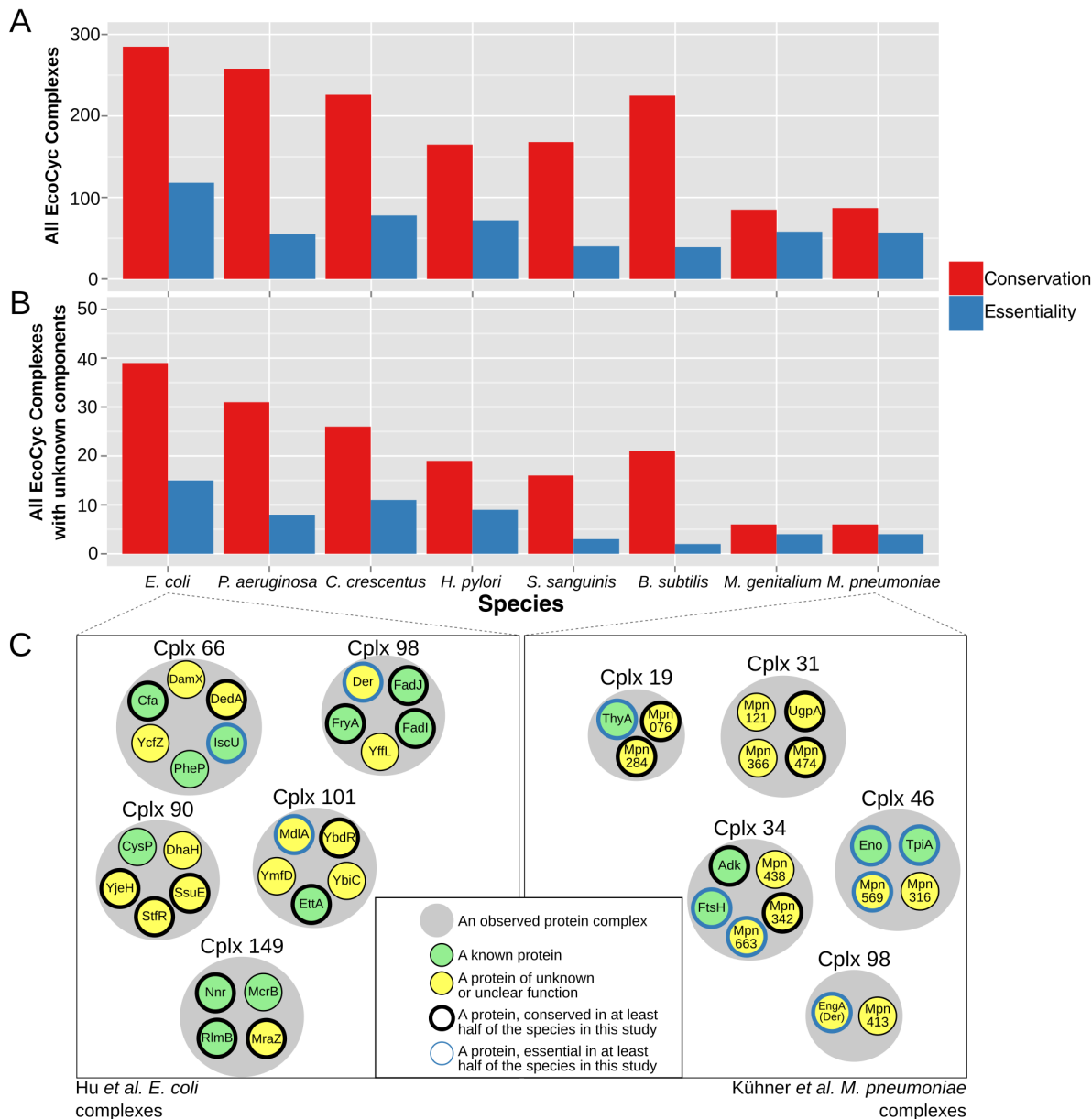


Fig. 2-Q. Protein complexes are rich in highly-conserved proteins of unknown function. (A) The list of EcoCyc *E. coli* protein complexes was compared on the basis of component presence vs. absence across seven other species in this study. Conserved complexes, in this figure, are those in which at least one orthologous component is present in the target species. Similarly, essential complexes include at least one component found to be essential in both *E. coli* and in the target species. (B) As in (A), but within the subset of EcoCyc complexes containing at least one protein of unknown or unclear function. In these instances, the complex itself may have a known function though the roles of its components may remain unclear. (C) Examples of experimentally-observed protein complexes containing proteins of unknown function. *E. coli* complex examples from Hu et al. are shown at left, *M. pneumoniae* complexes from Kühner et al. are shown at right. Complexes are labeled with the identifier used in their corresponding study.

Fig. 2-Q-C displays example complexes for the Hu (*E. coli*) and Kühner (*Mycoplasma pneumoniae*) complexes, respectively. Each complex contains at least one component of unknown or unclear function, whether in the context of the protein complex or broader cellular function. For instance, complex 66 from Hu et al. (**Fig. 2-Q-C**) consists of 6 unique proteins of which 3 are of unknown function (or remain without annotation). Of the 6 proteins, 3 are highly conserved and 1 of those three is frequently essential. The *E. coli* protein MraZ, present in Hu complex 149, is shown here as a protein of unknown function but was recently found to be a transcriptional regulator involved in multiple pathways (Liechti et al. 2012; Eraso et al. 2014). More than 149 Hu et al. *E. coli* complexes and 34 Kühner et al. *Mycoplasma pneumoniae* (183 in total) complexes contain at least one component of unknown function. Of these, 109 Hu et al. *E. coli* complexes and 19 Kühner et al. *M. pneumoniae* complexes contain components highly conserved as essential proteins.

2.4.7 The *E. coli* protein complex interactome

Interactors in this network are protein complexes from the EcoCyc set, filtered to remove all homomer complexes (that is, complexes only involving multiple copies of the same protein). The full set of complexes, with homomers, is 781 complexes; selecting only heteromers reduces the list to 285 complexes, as seen in the previous sections. These complexes were placed into one of eleven general functional categories (see Methods for details) based on their EcoCyc annotation.

With protein complexes serving as nodes of a network, we may then define interactions between complexes using the interactions between protein components of the complexes. Doing so yields a network of 217 nodes and 1,709 interactions. 68 complexes have not been observed to participate in cross-complex interactions as per available data and are not present in the network. The 801 interactions in the resulting network are filtered further by removing 908 self-interactions and 323 duplicate interactions (here, a duplicate indicates that two different protein pairs interact between the complexes), yielding a network of 210 complexes and 478 cross-complex interactions (**Fig. 2-R**, see also **Appendix Table III-L** for all interactions).

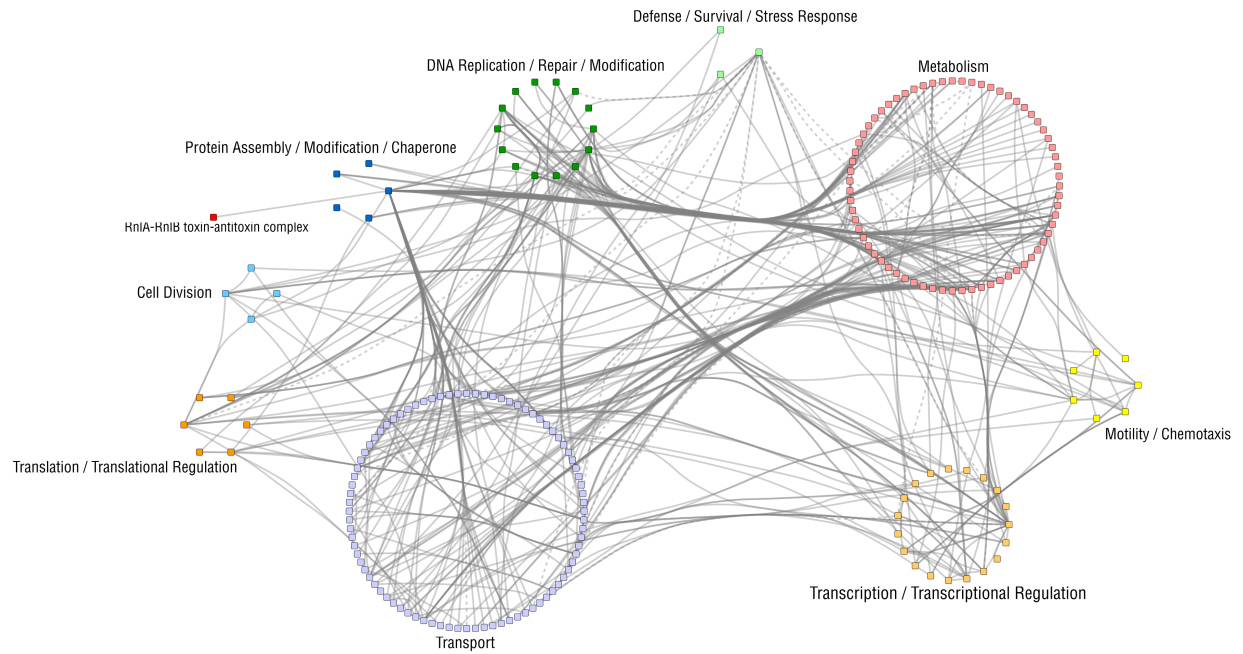


Fig. 2-R. Interaction network of *E. coli* protein complexes. Each node in this network is a single protein complex from *E. coli* as defined by EcoCyc. Interactions between complexes, denoted by edges, indicate that at least one protein component of the source complex has been observed in a protein-protein interaction with at least one protein component of the target complex, as per IntAct. Node color indicates general complex function. Edges have been bundled for clarity. Dotted line edges indicate orthology-based predicted interactions; all other interactions are based off direct interactions. Self-interactions are not shown.

The network contains one primary component with 206 of the 210 complexes and two separate components of one interacting complex pair each (including, specifically, the cytochrome *bd-I* and *bd-II* terminal oxidases). The network contains both direct interactions between *E. coli* proteins and those predicted from orthology (i.e., if two proteins are found to interact then all other proteins in their respective orthologous groups are predicted to participate in similar interactions). The overall network has a clustering coefficient of 0.172, a diameter of 7, and a density of 0.022. Each node has an average of 4.552 neighbors and the highest degree node (complex CPLX0-3934, the GroEL-GroES chaperonin complex) has a degree of 88, followed by the core RNA polymerase enzyme (complex APORNAP-CPLX) with a degree of 30.

2.4.8 Flexibility of protein complexes

The essential “core” components of protein complexes may be conserved across taxonomic levels while “accessory” components may not (Ryan et al. 2012). Given their multiple interactions, proteins within protein complexes should not only be more highly conserved than “un-complexed” ones, but should retain their essential roles if their fellow complex members are present (Hart et al. 2007; Wang and Marcotte 2010). Components of protein complexes are, on average, more likely to be present in other bacterial species than proteins not in complexes (Ryan et al. 2012). This is a result of high conservation among sets of large, essential complexes. 128 out of 285 literature-verified *E. coli* protein complexes appear to have all components conserved in *B. subtilis*, 30 of which also appear to be present in the *M. genitalium* genome. For

instance, all components of the ATP synthase complex (EcoCyc: ATPSYN-CPLX) are present in all species examined, though they do vary in essentiality. *B. subtilis* essentiality screens found no essential genes in ATP synthase, while those for *M. pneumoniae* found all but one component to be essential. Other complexes – predominantly those with transmembrane domains and/or transporter functions – are more variable in both conservation and essentiality, though they provide examples of how dispensable accessory proteins may be.

Some protein complexes with essential functions in *E. coli* may not be present in other species. The lipopolysaccharide transport complex (EcoCyc: CPLX0-7992) serves as an excellent example: all seven of the Lpt proteins in this complex have been found to be essential in *E. coli* though their conservation is limited to other Gram-negative species including *C. crescentus* and *P. aeruginosa*. I found that most transmembrane protein complexes follow this pattern, likely reflecting the link between speciation and environmental conditions as transmembrane transport complexes are necessary for obtaining resources from the environment. Interestingly, species with partial complex component conservation vs. *E. coli* may highlight situations in which core elements of a complex are conserved but have been modified to carry out other functions or adapted to special physiological circumstances. For example, 3 out of 4 of the succinate dehydrogenase complex (EcoCyc: SUCC-DEHASE) components in *E. coli* are also present in *B. subtilis* but not at all in *S. sanguinis*. This is an especially interesting

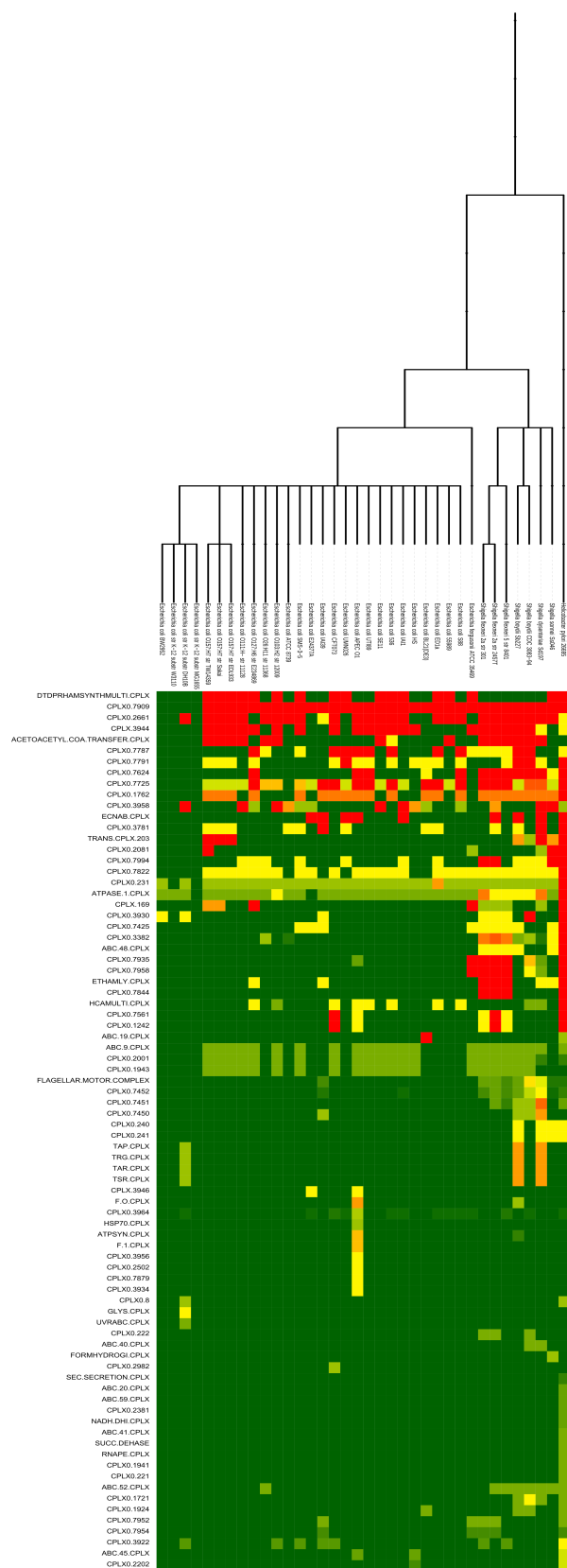
example as two of the components, SdhC and SdhD, are inner membrane proteins, though only SdhC is present in the three-component *B. subtilis* succinate dehydrogenase. I conclude that membrane proteins and their complexes are particularly malleable, given their role in signaling and transport which reflects adaptations to specific environments and the nutrients present in them.

Surprisingly, many essential proteins are poorly conserved and essentiality itself is often not conserved across species (**Fig. 2-I**). This suggests that many functions can be replaced by non-homologous displacement (Kelkar and Ochman 2013) and that genomes are more malleable in evolutionary terms than previously expected. Clearly, this evolutionary flexibility has contributed much to the success of microbes to populate all possible environments on the planet. Variability in complex conservation highlights a limitation with this study: I am unavoidably limited by the availability of sequenced bacterial genomes. Newly-characterized genomes may reveal additional variation or consistency among protein complexes even if they are highly reduced in other respects. As with their protein components, individual complexes reveal underlying evolutionary processes (**Fig. 2-L**). The most highly-conserved complexes are those with functions critical to microbial life, including transcription, translation, and transcript degradation. Though different RNA polymerase (RNAP) holoenzymes (that is, RNA polymerases with different sigma factors) were considered as distinct complexes in this study, all bacterial species unsurprisingly retained at least one type of RNAP. The ribosome (EcoCyc:

CPLX0-3964) is also well-conserved though its size and high level of conservation may obscure cross-species differences.

Variable conservation of some complexes is visible even among the *Escherichia* genomes (**Fig. 2-S**). CPLX0-7909 (the RnIA-RnIB toxin-antitoxin complex) only appears to be present in K-12 *E. coli* but also in single species of *Shewanella* and *Photobacterium*. This toxin-antitoxin system has a role in bacteriophage resistance in *E. coli* (Koonin et al. 1996) but it is unclear if this function may be retained in distantly related bacteria. CPLX0-2001 (the ferric dicitrate transport system) provides an example of more gradual change. This complex spans the membrane, suggesting its conservation should be membrane-dependent. This appears to be the case as it is well conserved across most Proteobacteria (except the *Rickettsiales* and *Buchnera* species) yet is poorly-conserved across most of the species traditionally considered Gram positive. A subset of complexes, including CPLX0-1163 (HslVU protease) and ABC-56-CPLX (aliphaticsulfonate ABC transporter), fit a strict co-conservation model: these complexes are almost always present in their full form rather than as a fraction of the *E. coli* model complex. These complexes are exceptions rather than the rule. Using *E. coli* as a model, few complexes are conserved perfectly across a wide range of species; in fact, most complexes are fractionally conserved.

Fig 2-S. All EcoCyc complexes and their fractional conservation in selected strains of *E. coli* and *Shigella*. Figure continues on next page.



EcoCyc Complex



A complex-centric approach to cross-species gene and protein conservation need not be restricted to *E. coli*. Rather, such an approach would benefit from expansion into other bacterial species as it may reveal novel evolutionary patterns and avoid the bias inherent in focusing on well-studied model bacterial species. The approach also offers a way to type potential bacterial pathogens: by comparing pathogenic strains with each other or those of other species, researchers may quickly identify not only the protein complexes implicated in pathogenesis (that is, those already suspected to be pathogenicity factors) but also unexpected gain or loss of other protein complexes or their components. Predicted protein complexes of *H. pylori* provide examples of conserved and missing protein complexes (**Fig. 2-T**). More than two thirds of the 1,180 distinct *H. pylori* orthologous groups encoded by its genome are present in *E. coli* as well (**Fig. 2-T-A**) suggesting that complex conservation should be similar (*M. pneumoniae* is included for comparison with a species distantly related to both *H. pylori* and *E. coli*). Reflecting both the evolutionary distance and difference in membrane structure between the two species, some *E. coli* protein complexes appear to be fully or mostly conserved (e.g., the OppBCDFMppA and DdpABCDF complexes) while others, such as the Bam outer membrane protein complex, are poorly conserved (**Fig. 2-T-B**). Unlike individual proteins, most protein complexes appear to be poorly-conserved from *E. coli* to *H. pylori* (**Fig. 2-T-C**).

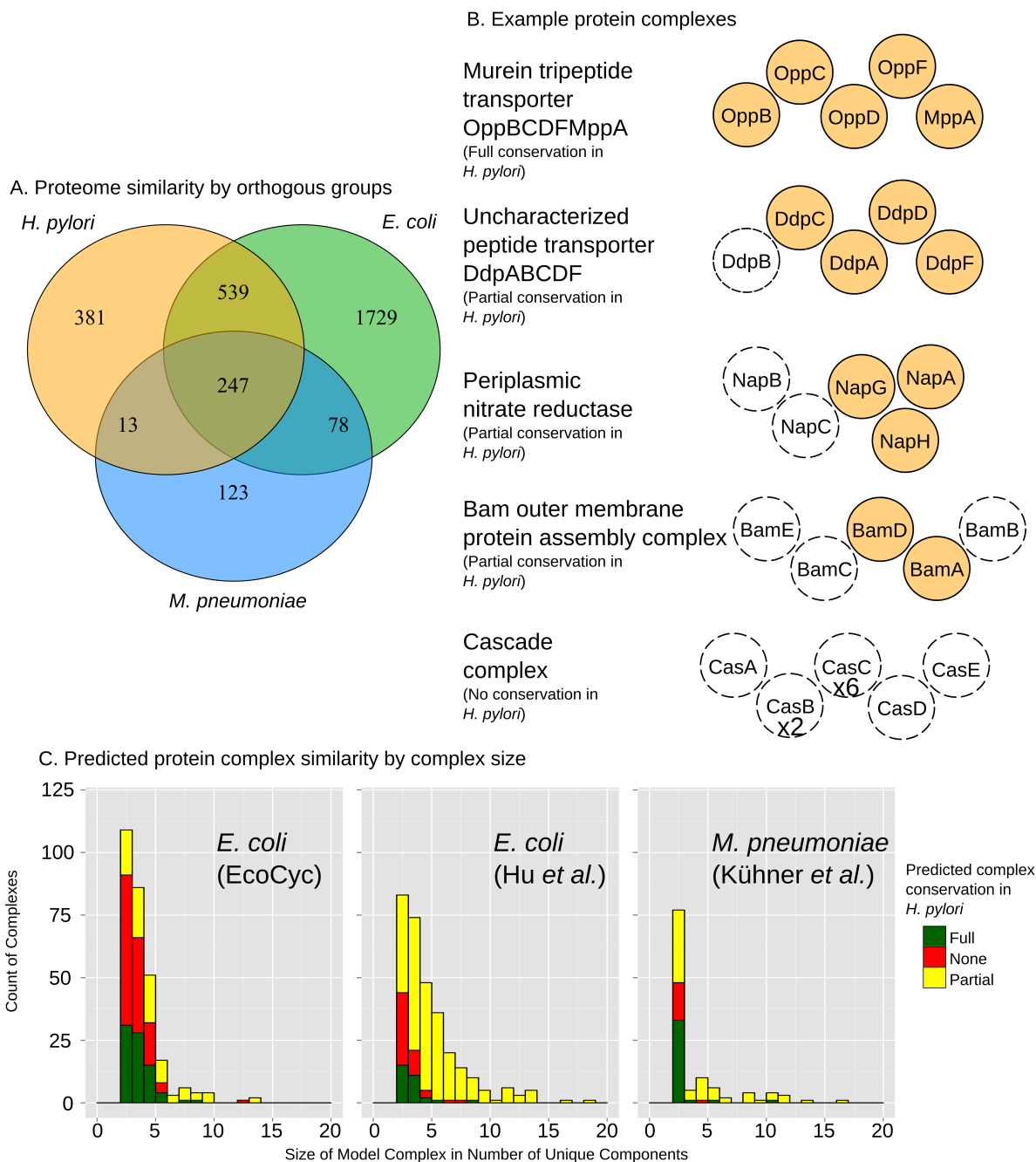


Fig 2-T. Predicted protein complexes in *H. pylori*. (A) Similarity of the proteomes of *E. coli*, *H. pylori*, and *M. pneumoniae* in terms of homologous proteins. (B) Selected protein complexes showing the variety of complex conservation. Dotted circles indicate proteins present in the model *E. coli* complex but predicted to be missing in *H. pylori*. Stoichiometry of complex subunits is noted where more than 1 protein component or multimer is present. (C) Similarity of predicted complexes when compared to *E. coli* and *M. pneumoniae*, the two other proteomes for which comprehensive protein complex information is available. Only complexes of 20 or fewer unique components are shown.

2.4.9 Further discussion

The substantial variation among protein complexes across species supports the notion that these complexes are much more malleable than previously thought. A possible explanation of this is that the function of a complex is more important than its content. Complexes can serve the same role yet contain different proteins and when one function is lost, others can fill in the gap. Other studies have found that functional redundancy can lead to variation and that there is little overlap in terms of protein interaction among species (Dixon et al. 2009; Ryan et al. 2013). While mutational change in a protein complex may have catastrophic potential, complexes are not immutable. In fact, several complexes that are essential in some species have varying composition in other species. For instance, 5 out of 9 components of the *E. coli* Sec translocation complex (EcoCyc: SEC-SECRETION-CPLX) are well-conserved across species from *P. aeruginosa* to *M. genitalium*. One of these components, SecA, has been found to be essential in all species focused on in this work with the exception of *S. sanguinis*; orthologs of this protein are present in all 894 bacterial genomes examined. The remaining 4 *E. coli* components are more variable in conservation across species. For instance, YajC is present in 727 out of the same 894 genomes. Strong selection pressure seems to avoid mutations that render the entire complex ineffectual. This may explain why I have observed a higher level of conservation for protein complex components than for proteins in general (**Fig. 2-D**).

Components of protein complexes are, on average, more likely to be present in other bacterial species than proteins not in complexes. The essential “core” components of protein complexes may be conserved across taxonomic levels while “accessory” components may not (Ryan et al. 2012). Given their multiple interactions, proteins within protein complexes should not only be more highly conserved than “un-complexed” ones, but should retain their essential roles if their fellow complex members are present (Hart et al. 2007; Wang et al. 2010). The amount of conservation is a result of high conservation among sets of large, essential complexes. 128 out of 285 literature-verified *E. coli* protein complexes are fully present in *B. subtilis*, 30 of which are also completely present in *M. genitalium*. For instance, all components of the ATP synthase complex (EcoCyc: ATPSYN-CPLX) are present in all species examined, though they do vary in essentiality. *B. subtilis* essentiality screens found no essential genes in ATP synthase, while those for *M. pneumoniae* found all but one component to be essential. Other complexes—predominantly those with transmembrane domains and/or transporter functions—are more variable in both conservation and essentiality, though they provide examples of how dispensable accessory proteins may be.

Some protein complexes with essential functions in *E. coli* may not be present in other species. The lipopolysaccharide transport complex (EcoCyc: CPLX0–7992) serves as an excellent example: all seven of the Lpt proteins in this complex have been found to be essential in *E. coli* though their conservation is limited to other Gram-negative species including *C. crescentus* and *P. aeruginosa*. I find that most transmembrane

protein complexes follow this pattern. Interestingly, species with partial complex component conservation vs. *E. coli* may highlight situations in which core elements of a complex are conserved but have been modified to carry out other functions or adapted to special physiological circumstances. For example, 3 out of 4 of the succinate dehydrogenase complex (EcoCyc: SUCC-DEHASE) components in *E. coli* are also present in *B. subtilis* but not at all in *S. sanguinis*. This is an especially interesting example as two of the components, SdhC and SdhD, are inner membrane proteins, though only SdhC is present in the three-component *B. subtilis* succinate dehydrogenase. I conclude that membrane proteins and their complexes are particularly malleable, given their role in signaling and transport which reflects adaptations to specific environments and the nutrients present in them.

Smaller and more reduced bacterial genomes (that is, relative to *E. coli*) appear to code for a greater fraction of highly-conserved protein complexes. This conservation is evident in comparisons of the *Mycoplasma pneumoniae* protein complexes. In an examination of these protein complex components across more than 800 bacterial genomes, I find that species such as *M. pneumoniae* offer a better model of the protein complexes most critical to bacterial life, though they lack the predictive power of protein complexes from *E. coli*. Protein complexes observed in *M. pneumoniae* may not only have retained a core set of functions but may also reflect a higher degree of multifunctionality among metabolic enzymes (Yus et al. 2009; Kelkar and Ochman 2013). In short, they may distribute more functions among fewer complexes. A truly all-

encompassing comparison of protein complexes should therefore incorporate multiple complex functions. In the absence of this data, however, model protein complex sets provide useful context for the relationships between diverse bacterial species.

Chapter 3 - Conservation of Protein-Protein Interactions among Bacteria

Significant portions of this chapter have been published or are in preparation for submission:

Caufield, J.H., Sakhawalkar, N., Uetz, P. (2012). A comparison and optimization of yeast two-hybrid systems. *Methods*, 58(4), 317–324. doi:10.1016/j.ymeth.2012.12.001.

Caufield, J.H., Wimble, C., Abreu, M., Shary, S., Wuchty, S., Uetz, P. (2016). Bacterial protein meta-interaction networks reveal consistencies among interactomes.

Manuscript submitted.

3.1 Abstract

Proteome-wide interactomes can offer compelling evidence for protein function. The protein interactomes of several bacterial species have been completed, including several from prominent human pathogens. In this study, I use more than 52,000 unique protein-protein interactions (PPIs) across 349 different bacterial species to determine their conservation across data sets and taxonomic groups. When proteins are collapsed into orthologous groups (OGs) the resulting meta-interactome still includes more than 43,000 interactions, about 14,000 of which involve proteins of unknown function. While conserved interactions provide support for protein function in their respective species data, I found only 429 PPIs conserved in two or more species. The meta-interactome serves as a model for predicting interactions, protein functions, and even full interactome sizes for species with limited to no experimentally observed PPI, including *Bacillus subtilis* and *Salmonella enterica* which are predicted to have up to 18,000 and

31,000 PPIs, respectively. Such conserved interactions should provide evidence for important but yet-uncharacterized aspects of bacterial physiology and may provide targets for anti-microbial therapies.

3.2 Introduction

Our understanding of a protein's role in a biological system strongly depends on its placement in a network of protein-protein interactions, or interactome. Recently, interactome data sets involving proteins from various microbial species have been constructed using experimental and inferred data (**Table 3-A**) while numerous databases have been created to store and disseminate this information (Kerrien et al. 2012, Chatr-Aryamontri et al. 2015, Szklarczyk et al. 2015). Bacterial proteomes are particularly attractive subjects for interactome analysis due to their manageable size. The proteomes of many bacterial species include only a few thousand proteins, suggesting that they are about an order of magnitude smaller than their counterparts in many animals and plants. Therefore, most bacterial species provide more tractable interactomes compared to the human genome that has more than 20,000 protein coding genes (ENCODE Project Consortium 2012) and more than 650,000 predicted protein interactions (Stumpf et al. 2008).

Table 3-A. Experimental microbial interactome sizes.

Species Name	Experimental interactome size (PPIs)	Proteins (and unique OGs) in proteome	Proteins in published interactome	Interactome ref.
<i>Campylobacter jejuni</i>	11,687	1,623 (1,523)	1,321	(Parrish et al. 2007)
<i>Escherichia coli</i>	2,234	4,306 (2,563)	1,269	(Rajagopala et al. 2014)
<i>Helicobacter pylori</i>	3,004	1,553 (1,280)	739	(Häuser et al. 2014)

<i>Mesorhizobium loti</i>	3,121	7,272 (2,981)	1,804	(Shimoda et al. 2008)
<i>Synechocystis</i> sp. PCC 6803	3,236	3,575 (2,246)	1,920	(Sato et al. 2007)
<i>Treponema pallidum</i>	3,649	1,036 (736)	726	(Titz et al. 2008)
<i>Saccharomyces cerevisiae</i>	957 - 37,600*	6,721 (4,794)	~1,004 - 3,000	(Uetz et al. 2000; Ito et al. 2001; Yu et al. 2008; Sambourg and Thierry-Mieg 2010)

Yeast (*Saccharomyces cerevisiae*) interactome sizes provided for comparison.

* Sambourg and Thierry-Mieg (2010) estimated the yeast interactome size to be ~37,000 PPIs, based on 3,042 PPIs among well-studied proteins curated from the literature.

Nearly all published bacterial interactomes have been created using either the yeast two hybrid (Y2H) system or affinity purification followed by mass spectrometry analysis (AP/MS). Although *E. coli* is the only bacterial species with a comprehensive interactome that has been studied by both Y2H (Rajagopala et al. 2014) and AP/MS (Hu et al. 2009) methodologies a comparison of both methods surprisingly showed largely non-overlapping interaction data sets. In the Y2H data set of 2,234 *E. coli* protein-protein interactions, 1,800 were found outside of known protein complexes (Rajagopala et al. 2014). Similarly, only a third of ~1,500 interactions that are thought to occur in protein complexes were detected by the Y2H approach, indicating that existing methodologies in isolation produce incomplete datasets (Rajagopala et al. 2014).

A way to overcome such problems is to combine not only different datasets from the same species but also data from different species. Although cross-species interactome approaches have been recently presented for human and yeast protein sets (Zhong et al. 2016) no comprehensive comparison of bacterial interactomes currently exists. While

the majority of reports focus on one interactome (**Fig. 3-A**), far fewer include data from more than one set of interactions, and just two recent reports have investigated more than 5 out of 11 available large-scale bacterial interactome studies. One of these studies provides an analysis of bacterial genomes in terms of their predicted functional complexity rather than the exact interactions in their interactomes (Kelkar and Ochman 2013). Other studies dealt with four or five published interactomes (see a complete list in **Appendix Table IV-A**), presenting only a general discussion of the evolution of protein networks (Ratmann et al. 2009) or a review of ways to mine high-throughput experimental data to link gene and function (Blaby-Haas and de Crecy-Lagard 2011).

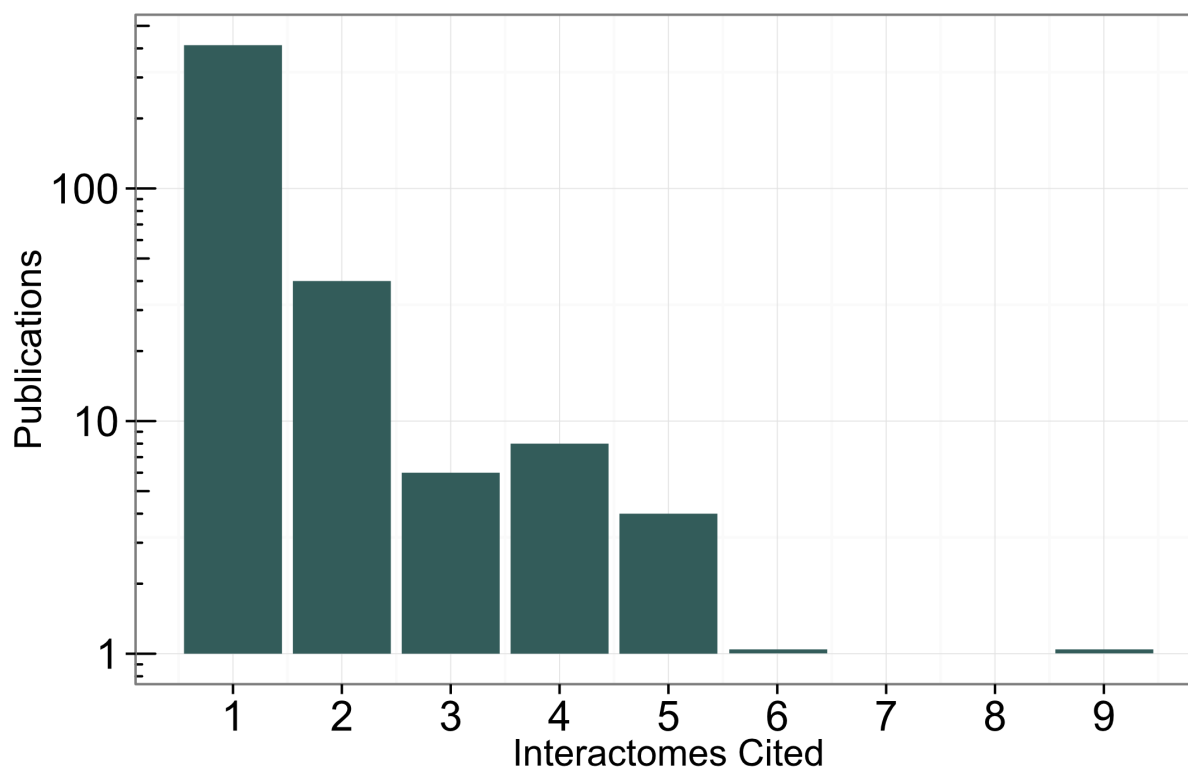


Fig. 3-A. Analysis of citations of bacterial protein interactome literature. Publications are any individual scholarly works indexed in PubMed Central with a reference to at least one of eleven large-scale bacterial interactome studies. Publications count is on a log scale. See Methods for details of citation analysis. Christopher Wimble assisted with this analysis.

One of the most promising applications of interactomics is in the analysis of protein function. In a “guilt by association” approach (Schauer and Stingl 2009), protein interactions provide context to proteins by considering functional roles of their known interaction partners. For example, a protein that interacts predominantly with proteins that participate in metabolic activities putatively has such functions as well. This method has been applied as part of the analysis of interactomics data (Titz et al. 2008, Hu et al. 2009, Song and Singh 2009) and is a major focus of interaction databases such as STRING (Szklarczyk et al. 2015) and BioGRID (Chatr-Aryamontri et al. 2015).

As part of a guilt-by-association approach, proteins within interaction networks may be compared by defining them as members of orthologous groups (OGs). The concept of OGs began with clusters of orthologous groups as defined by Tatusov et al. (1997) for the explicit purpose of allowing functional information about one member of a COG to apply to all other members of the cluster. Assembling these groups on the basis of sequence alone permits them to be used to infer functional contexts for gene and protein sequences without experimental characterization. The basic concepts defined by Tatusov et al. (1997) and other work by the Koonin group (Tatusov et al. 2003; Kristensen et al. 2010) have since been improved upon by databases such as eggNOG (Huerta-Cepas et al. 2016). The OGs in eggNOG (also referred to as NOGs, or *Non-supervised* orthologous groups) are defined through non-supervised, taxonomy-limited methods and (as of v.4.5, the most recent release) incorporate sequences from more than 1,600 prokaryote genomes. In this work, I use eggNOG OGs to reduce the

complexity of interaction networks by joining proteins of similar sequence and likely similar function.

An orthology-based approach may be species-independent and can allow interaction networks of different species to be used to predict uncharacterized, conserved interactions. Conserved, cross-species interactions may be referred to as “interologs” (Matthews et al. 2001). Comparing interactions derived from multiple screens and species can also provide an evolutionary basis for the reasons an interaction may or may not be present. Analyses of conserved networks have been performed (Matthews et al. 2001; Sharan et al. 2005; Liang et al. 2006; Wiles et al. 2010; Ryan et al. 2012), in some cases alongside interactome studies (Wang et al. 2010). Several studies have also attempted to assemble comprehensive interaction networks using orthology-based predictions (Brown and Jurisca 2005; Lee et al. 2008; Gu et al. 2011) or predictions based on physical protein properties (Zhang et al. 2012; Kotlyar et al. 2014). Few of these studies – with the exceptions of Sharan et al. (2005) Zhang et al. (2012) – have incorporated interaction data from bacteria and none have incorporated data from as many species as those used in this work. Most of this previous work has also been limited by low proteome coverage in the underlying interactomes or relies upon gene essentiality data which, as shown in Chapter 2, is frequently inconsistent across different species.

Here, I combine experimentally-derived, previously published protein-protein interactions from 349 bacterial species to form a consensus meta-interactome. This approach uses orthologous groups (OG) of proteins to combine all known interactions into a single network. Notably, I observe that such a network shares characteristics of single species interactomes. Furthermore, the augmentation of single species interaction networks with a bacterial meta-interactome boosts its ability to predict functions of the underlying proteins, given its dramatically increased information content. Finally, I use such a bacterial meta-interactome to predict interactome sizes of species for which only incomplete interaction data is available.

3.3 Experimental methods

3.3.1 PPI detection assay comparisons

Prior to much of the work presented in this chapter, I compared the results from several variations on protein-protein interaction screens to determine the extent to which methodological differences impacted interaction detection. These analyses are available in Caufield et al. (2012); the comparison methods are as follows.

I used three primary datasets for analysis. The interactions among human proteins used by Braun et al. (2009) were originally selected from detailed small-scale studies and subsequently systematically retested (Chen et al. 2010). Here, I reanalyzed the raw data from the Chen dataset, finding slightly different numbers than were originally reported (here, I counted all yeast colonies that grew to above background levels in at

least two of four colonies per plate and on at least one of the two plates used). The interactions of both Varicella Zoster Virus (Stellberger et al. 2010) and phage lambda (Rajagopala et al. 2011) proteins were also included in this analysis as published. Unlike many other sets of published protein–protein interactions, these datasets have been systematically generated by use of four different Y2H vectors, as detailed by Stellberger et al. (2010).

The aggregate results from each method used by Braun et al. and Chen et al. were compared by clustering to determine how similar the detected subsets of the reference set are. The results of all assays from both studies were treated as an array of 92 weighted values. Each result for a specific PPI within the positive reference set (PRS) and random reference set (RRS), both as defined by Braun et al. (2009), was treated as a single value with positive results holding a maximum value of 1 and negative results holding a value of 0. All PPIs reported by Braun et al. were assigned a value of 1, as the exact number of replicates performed in these assays is unclear. All PPIs observed in the Chen et al. dataset were assigned a weighted value as follows: if a PPI was observed for all replicates at a 3-AT concentration of 0, 3, or 10 mM, they were assigned a value of 0.1, 0.4, or 0.5, respectively. PPI observed in only 1 of 2 replicates at the same 3-AT concentrations were assigned half of the full values, for 0.05, 0.2, or 0.25, respectively. The weighted values for all three 3-AT concentrations were added for each PPI in the PRS and RRS, such that the results for each vector combination could be

treated as an aggregate of stringency and replication, with greater values for PPI observed at multiple stringency levels and in multiple replicates.

All results arrays were aligned and clustered using the PermutMatrix graphical data analysis package (Caraux and Pinloche 2005). Hierarchical clustering was performed by unweighted pair group method with arithmetic mean (UPGMA) and Euclidean distance to reflect similarities within the assay data.

3.3.2 Literature mining for citation analysis

The initial stages of this project required assessment of whether comparisons of bacterial protein-protein interactomes were common in the interactome literature. A list of 11 publications, each describing a single bacterial protein-protein interactome, was assembled as a representative set of the bacterial protein-protein interactome literature. The publications and their corresponding foci are listed in **Table 3-B**.

Table 3-B. Set of comprehensive bacterial protein interactome studies used for citation analysis.

Reference	Species of Focus
Cherkasov et al. (2011)	<i>Staphylococcus aureus</i>
Häuser et al. (2014)	<i>Helicobacter pylori</i>
Hu et al. (2009)	<i>Escherichia coli</i>
Kühner et al. (2009)	<i>Mycoplasma pneumoniae</i>
Parrish et al. (2007)	<i>Campylobacter jejuni</i>
Rain et al. (2001)	<i>Helicobacter pylori</i>
Rajagopala et al. (2014)	<i>Escherichia coli</i>
Sato et al. (2007)	<i>Synechocystis</i> sp. PCC6803
Shimoda et al. (2008)	<i>Mesorhizobium loti</i>

Titz et al. (2008)	<i>Treponema pallidum</i>
Wang et al. (2010)	<i>Mycobacterium tuberculosis</i>

The full list of citations from each paper was retrieved from PubMed Central in XML format in August 2015. All citation lists were combined to determine citations shared by multiple publications in the set. Publications citing multiple representative interactome publications are those with potential for cross-interactome comparisons. See **Appendix Table IV-A** for full citations for each publication in the set.

3.3.3 Protein interaction data sets

Protein interaction sets were obtained and filtered using an in-house Python program, *Network_umbra* (available at <http://github.com/caufieldjh/network-umbra>). This program parses interaction data files in PSI-MI TAB 2.7 format (MITAB27; a format used by protein-protein interaction databases; developed by the HUPO Proteomics Standards Initiative and described in detail at <https://code.google.com/p/psimi/wiki/PsimiTab27Format>) and facilitates all further methods described in this study.

The full set of interactions was obtained from the IntAct database (Kerrien et al., 2011; <http://www.ebi.ac.uk/intact/>) on September 25, 2015. To produce the data set used in this study, the full set of IntAct interactions was filtered by Uniprot taxonomy to include only protein-protein interactions (PPI) from bacterial sources (species:"taxid:2"). Prior to further filtering, this interaction set includes 63,421 interactions across all interaction

types. All interactions without Uniprot identifiers (i.e., interactions involving ChEBI chemicals) were removed, as were interactions with erroneous annotation (i.e., interactions involving bacterial proteins vs. eukaryote proteins). The set of IntAct interactions was appended with the protein interactome of *Mesorhizobium loti* (Shimoda et al. 2008). Where possible, proteins were assigned membership in orthologous groups (OGs) using eggNOG v.4 NOGs (Powell et al., 2014, Huerta-Cepas et al. 2015); proteins without OG annotation are treated as single-member OGs and referred to using their UniprotAC identifiers. All protein-protein interactions are retained in the data set regardless of experimental observation method; interactions derived from spoke-expansion models are treated identically to those defined as “direct” interactions.

3.3.4 Construction of meta-interactome networks

The full set of protein interactions sourced from IntAct as described above constitutes the starting data set for meta-interactome construction. In this study, I define a meta-interactome as a set of protein-protein interactions where similar proteins and the interactions among those proteins are merged into single interactor groups and interactions (**Fig. 3-B**). Interactions among proteins of the same group are considered a self-interaction, though all interactions retain properties of the source interaction network, including the count of protein interactions and count of unique source species contributing to the interaction. Meta-interactome groups are defined by eggNOG v.4 NOGs as noted above. Because annotations for interactions involving similar proteins from closely-related species may differ, the species and strains corresponding to each

interaction were labeled using NCBI Taxonomy identifiers and identifiers sharing a parent or a child were merged. All interactions were compressed using OG-annotated proteins such that each OG-OG interaction appears in each data set only once per species, though a protein may belong to multiple OGs (in these cases, the resulting OG name includes both identifiers separated by a comma, e.g. "COG1100,COG4886").

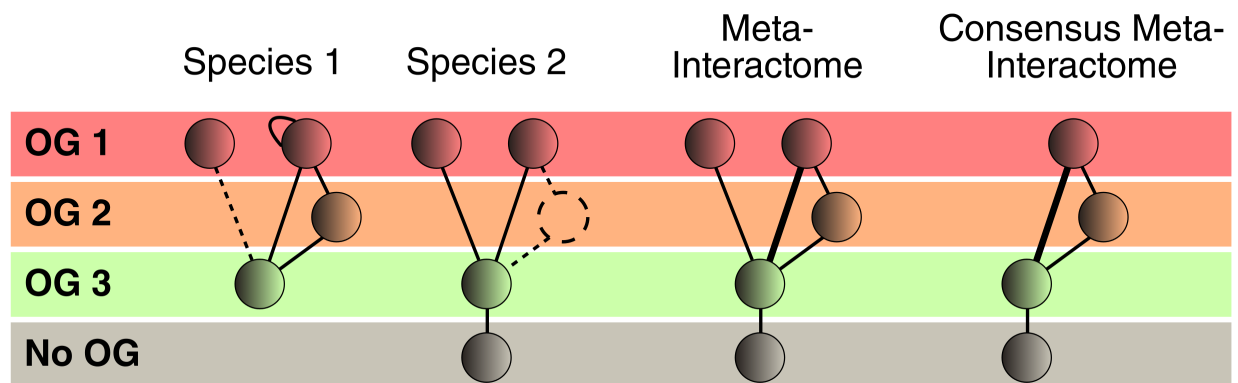


Fig. 3-B. Concept and construction of the meta-interactome. The set of interactions between proteins in different bacteria is defined by a meta-interactome. The meta-interactome is constructed by joining interactions between proteins from multiple species such that single proteins in shared orthologous groups (OGs) are treated as single nodes in the meta-interactome network. The meta-interactome is compressed further into a consensus meta-interactome by treating *all* proteins as OGs, even when multiple proteins from the same species share an OG. As shown in this figure by edge thickness between nodes, interactions seen in multiple species provide a weighted value to interactions in the meta-interactome networks.

The full meta-interactome is provided in **Appendix Table IV-B** in PSI-MI TAB 2.7 format, with the addition of orthologous groups in the final two columns (corresponding to interactors A and B, respectively). This interactome contains 52,734 interactions among 12,706 unique proteins, 1,805 (3.4%) of which fail to map to an orthologous group. Treated as a network of OGs, this network contains 8,521 unique interactors.

A further subset of the meta-interactome was prepared such that this set merged all interactions on the basis of shared interactors (see **Appendix Table IV-C**). For example, two different interactions between proteins in OG1 and proteins in OG2 are considered a single interaction. Furthermore, each OG-OG interaction is counted as a single interaction across any number of species. I refer to this set as the *consensus meta-interactome*. This network contains 8,475 unique interactors and 43,545 interactions.

Network construction, visualization, and analysis was performed using Cytoscape (Shannon et al., 2003) v.3.

3.3.5 Interactome size prediction

The program *Network-umbra* is a set of Python scripts that uses the consensus meta-interactome of OG-OG interactions to generate predicted interactomes for a given bacterial species. See **Appendix II** for a full guide to this software. Given a list of

UniprotAC identifiers, *Network-umbra* assigns each to an OG and constructs a set of interactions among those OGs based on their presence in the consensus network. In most cases, predictions are general and unverified: if a pair of OGs is present in the consensus network they are predicted to interact in any context. Reference proteomes used for interactome size prediction were retrieved from Uniprot on July 20, 2016 and are listed in **Table 3-C**.

Table 3-C. Reference proteomes used for interactome size prediction.

Species and Strain	NCBI Taxonomy ID
<i>Bacillus subtilis</i> str. 168	224308
<i>Caulobacter crescentus</i> CB15	190650
<i>Escherichia coli</i> K-12	83333
<i>Helicobacter pylori</i> 26695	85962
<i>Mesorhizobium loti</i> MAFF303099	266835
<i>Mycoplasma genitalium</i> G37	243273
<i>Pseudomonas aeruginosa</i> PAO1	208964
<i>Salmonella enterica</i> subsp. <i>enterica</i> serovar Typhi	90370
<i>Staphylococcus aureus</i> subsp. <i>aureus</i> NCTC 8325	93061
<i>Synechocystis</i> sp. PCC 6803 substr. Kazusa	1111708
<i>Treponema pallidum</i> subsp. <i>pallidum</i> str. Nichols	243276

3.3.6 Functional annotation

Interactors in the meta-interactome are annotated using the functional categories originally used by Tatusov et al. (1997) for the COG project and by eggNOG (Huerta-Cepas et al. 2015). They are listed in **Table 3-D**.

Table 3-D. Functional categories used to describe orthologous groups of bacterial proteins.

Category Letter	Category
A	RNA processing and modification
C	Energy production and conversion
D	Cell cycle control, cell division, chromosome partitioning

E	Amino acid transport and metabolism
F	Nucleotide transport and metabolism
G	Carbohydrate transport and metabolism
H	Coenzyme transport and metabolism
I	Lipid transport and metabolism
J	Translation, ribosomal structure and biogenesis
K	Transcription
L	Replication, recombination and repair
M	Cell wall/membrane/envelope biogenesis
N	Cell motility
O	Posttranslational modification, protein turnover, chaperones
P	Inorganic ion transport and metabolism
Q	Secondary metabolites biosynthesis, transport and catabolism
T	Signal transduction mechanisms
U	Intracellular trafficking, secretion, and vesicular transport
V	Defense mechanisms
W	Extracellular structures
S	Function unknown

3.4 Results and discussion

3.4.1 The bacterial meta-interactome resembles individual interactomes in structure

A direct comparison of interologous protein-protein interactions benefits from increasing the amount of data used. Though complete interactome sets are ideal for this purpose, even a comprehensive, proteome-wide interactome does not contain every biologically-relevant PPI. In the case that the missing PPIs are not present for methodological or sequence-related reasons (i.e., if two proteins conserved across multiple species produce a detectable interaction in just one species) data sets produced using different

methods and/or proteomes will complement each other. I constructed a meta-interactome of individual bacterial protein interactomes to address these concerns and to determine how useful such a data aggregate could be.

To compare interactions across multiple species, I first mapped proteins to orthologous groups (OGs; for details see Materials and Methods). As a source of information about OGs, I used the EggNOG database (Huerta-Cepas et al. 2016), expanding the idea of clusters of orthologous groups (Tatusov et al. 2000) constructed from numerous organisms. As a source of protein interactions in bacteria I used the IntAct database (Kerrien et al. 2012). Furthermore, I included the protein interactome of *Mesorhizobium loti* (Shimoda et al. 2008), a protein interaction data set not available in the IntAct database at the time. Accounting for all experimental sources of protein interactions, I found that the majority of interactions (> 60%) have been found in *E. coli* and *C. jejuni* (**Fig. 3-C**). Based on the total set of roughly 52,000 interactions between proteins in the underlying organisms, I connected interactors sharing OGs and used the number of species found to share an interaction as weighting value for each interaction.

In total, I obtained a consensus meta-interactome of 8,475 orthologous groups embedded in web of 43,545 weighted links, covering 349 distinct bacterial species (**Fig. 3-D-A, B**; see **Appendix Table IV-B** for details). Such a network consists of 205 connected components and includes 1,352 self-connected nodes. Moreover, the largest component pooled 88.9% of all nodes. I observe that the majority of OGs in the meta-

interactome correspond to a single protein interactor while the majority of links is composed of one interaction. As the average weight of links is 1.0 ± 0.1 , I can consider the network to be largely unweighted. As a consequence, I find that the average path length in the unweighted network is 3.7 ± 0.9 while the diameter of the network is roughly 15, indicating small world network characteristics (Gallos et al. 2012). The average number of neighbors is 10.2 ± 23.9 ; this average is likely influenced by the presence of several broadly-defined OGs. (I interpret interactors with a shared OG to be potential paralogs due to their sequence similarity, so true paralogs participating in similar interactions will likely be present in the same group). Demonstrating the scale-free tendency of many similar networks (Reed 2008), I found that the distribution of the number of neighbors decays as a power-law (**Fig. 3-E**).

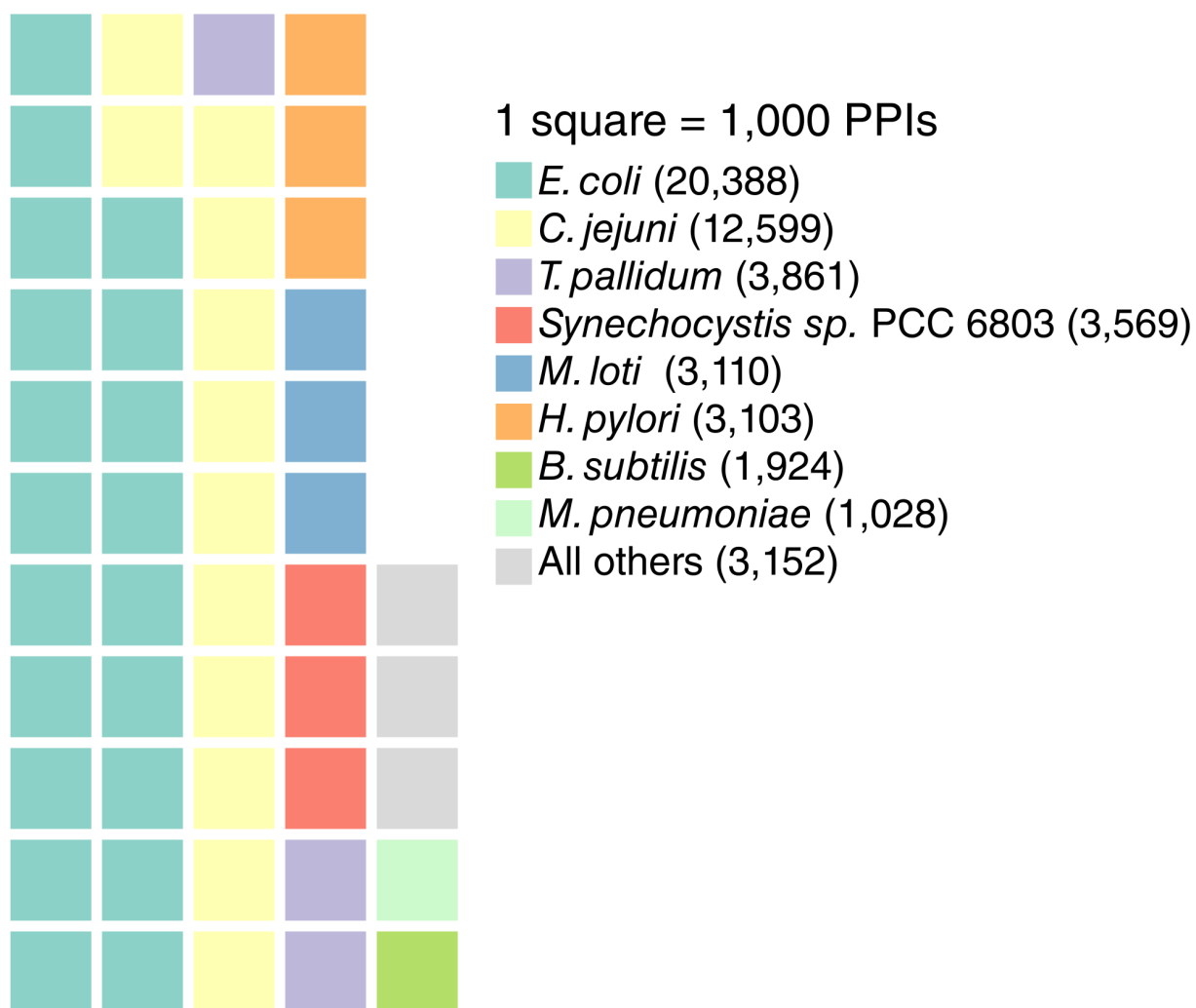


Fig. 3-C. Composition of the meta-interactome. A breakdown of source species of the meta-interactome. Protein interactions in *E. coli* and *C. jejuni* contributed to more than half of the total set of interactions in the meta-interactome.

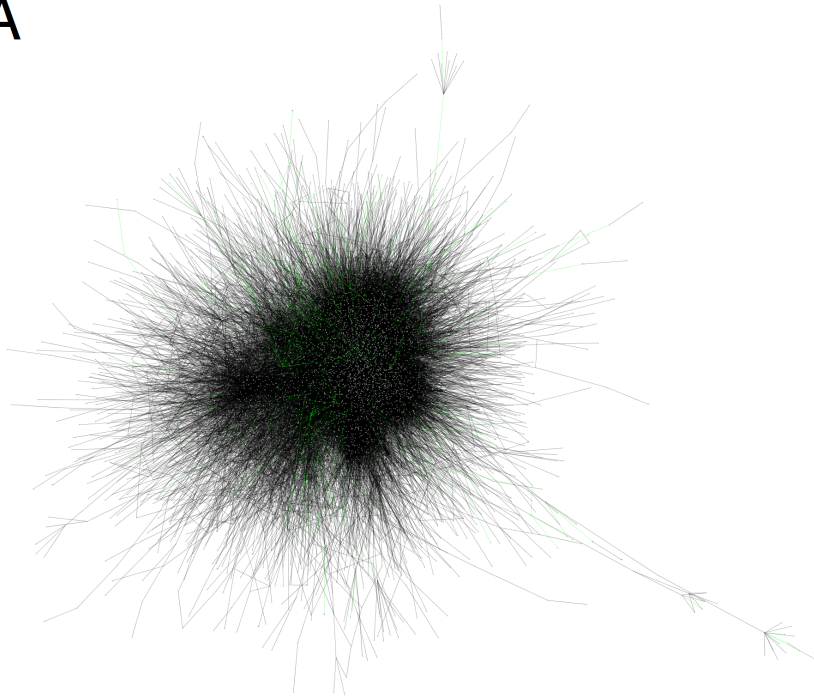
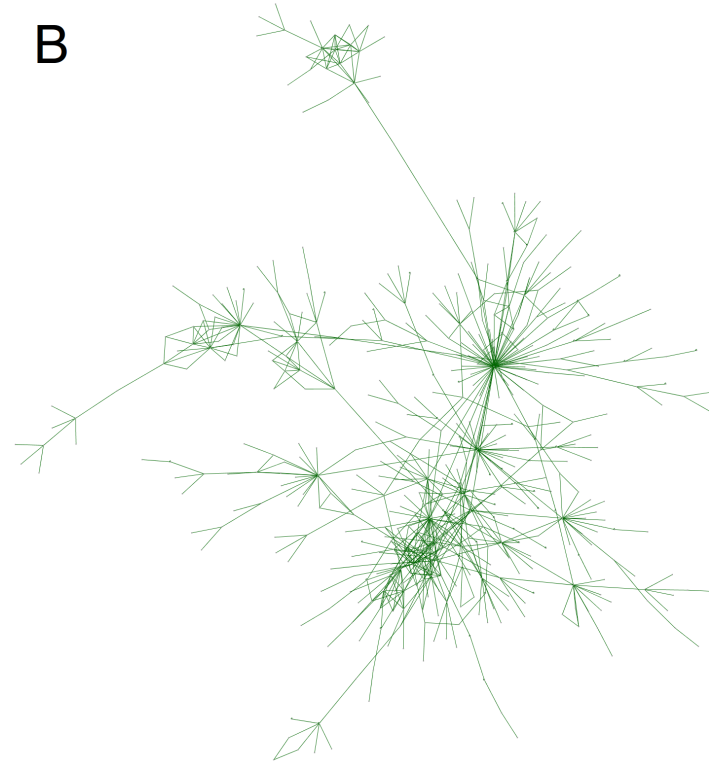
A**B**

Fig. 3-D. Overall structure of the main component of the consensus meta-interactome. A) All interactions of the main component of the consensus meta-interactome are shown. This segment of the network includes 7,373 nodes and 42,098 edges, or 86.9% of all interactors and 96.7% of interactions in the network. Interactions with more than two PPIs contributing to their interaction in this network are highlighted in green; these include 986 nodes and 1,186 edges. In this component, each node has an average of 11.138 neighbors. The average count of of interactions contributing to each edge shown here is 1.18 ± 0.922 . **B)** The network subset of interactions highlighted in part A. The main component of this subset, including 499 nodes and 784 edges, is shown, with all other interactions omitted for clarity. This subset is composed of edges derived from the most frequently observed types of interactions in the meta-interactome. In the full subset, including interactions not shown here, each node has an average of 2.004 neighbors and the average count of interactions contributing to each edge is 4.77 ± 3.80 .

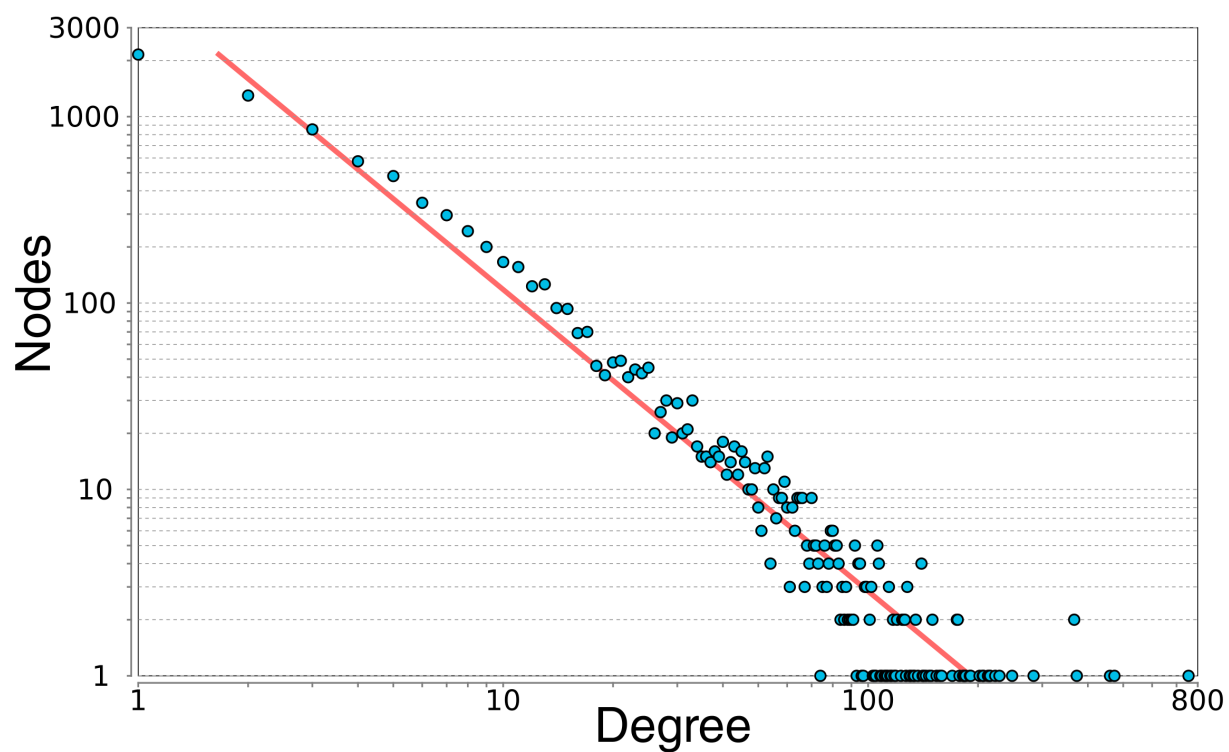


Fig. 3-E. Properties of the consensus meta-interactome. The number of neighboring OGs decays according to a power law ($R^2 = 0.907$) in the consensus meta-interactome.

At this point, OGs serve as nodes and any interaction between proteins of one OG and those of another provides an edge between nodes. The network contains 8,475 nodes and 43,545 edges, with 349 distinct bacterial taxids contributing interactions. There are 168 unconnected nodes as well as 1,352 self-connected nodes. The average path length is 3.828 and the average number of neighbors is 9.957. This number is significantly larger than the average of ~3.8 interactions per protein across the six experimental bacterial interactomes (**Table 3-A**). Even the two networks with arguably the highest number of false positives, *C. jejuni* and *T. pallidum*, have average degrees of 8.8 and 5.0, respectively, much less than those with the lowest value, about 1.7, in *Synechocystis*, *Mesorhizobium*, and *E. coli* (**Table 3-A**).

The meta-interactome has a network diameter of 15 which indicates that it is a small world network (Gallos et al. 2012). The majority (~88.9%) of the nodes in this network form a single, interconnected component (**Fig. 3-D**) though most nodes are connected by just a single interaction. Though, as mentioned above, the degree distribution follows a power law – demonstrating the scale-free tendency of many similar networks (Reed 2008) – in this case the distribution may be the result of nodes representing just a single protein. In biological terms, this signals a limited potential for interaction compared to those OGs containing many proteins.

The majority of the interactions in the consensus meta-interactome are contributed by a small set of species. More than half of the interactions are found in either *Escherichia coli* or *Campylobacter jejuni* (16,276 interactions and 11,308 interactions, respectively, or more than 63% of the interactome in total, though some interactions are seen in both species). An additional five species (*Treponema pallidum*, *Synechocystis* sp. PCC 6803, *Mesorhizobium loti*, *Helicobacter pylori*, and *Bacillus subtilis*) each contribute more than a thousand additional interactions. Each of these species, with the exception of *B. subtilis*, has been the subject of a comprehensive protein-protein interactome study. While all of these species are also well-studied, their contributions to the meta-interactome are a combination of data from single interactomes and those from smaller, more focused studies.

3.4.2 Functional annotation of orthologous groups and their conservation

Single interactomes are known to have many gaps, i.e. interactions that went undetected in experimental studies (Friedel and Zimmer 2006, Guimera and Sales-Pardo 2009). Since a missed interaction in one study may be found in an independent study through evolutionary conservation of the corresponding proteins, a meta-interactome network potentially reveals such gaps. As such, I assume that links between orthologous groups in the consensus meta-interactome may be indicative of undetermined interactions between orthologs in the corresponding organisms. Counting the number of bacteria a given interaction was observed in, I found that relatively few interactions appear in multiple bacterial species (**Fig. 3-F**). In particular, I found 43,116

interactions occurred only in a single species, 361 appeared in two species while only 68 interactions occurred in three or more species.

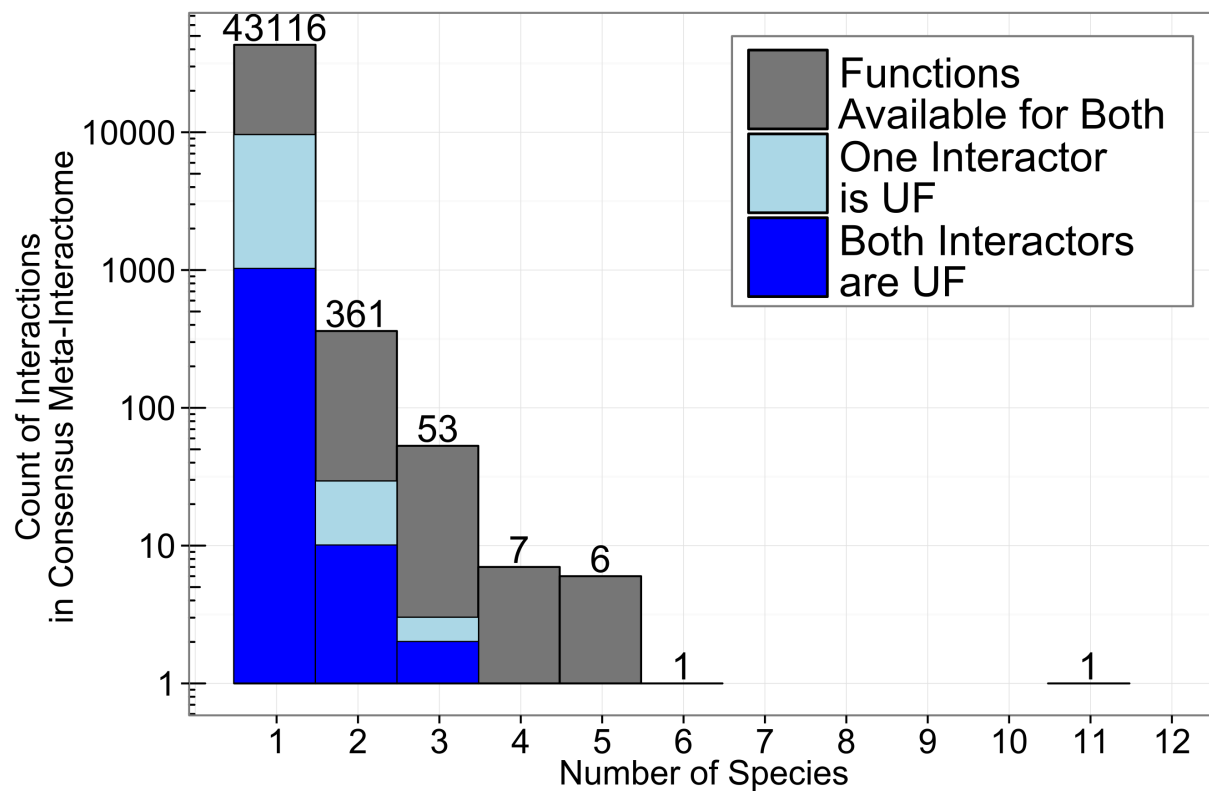


Fig. 3-F. Conserved interactions in the consensus meta-interactome. Counts of interactions in the consensus meta-interactome network. Nspecies indicates the number of distinct bacterial species contributing the interaction; a value of 1 denotes an interaction observed for a single species only. For each count, subsets denote how many interactions involve two, one, or zero interactors of known function (as both, one, and none, respectively).

Any single bacterial proteome may contain hundreds or even thousands of proteins of unknown or unclear function. Out of more than 43,000 interactions, fewer than 10,000 involve two interactors of unknown or unclear function (**Fig. 3-F**). Due to limited cross-species overlap, just a small subset of fewer than 100 interactions is observed in more than one species and involves one or more interactors of unknown function. The limited extent of cross-species overlap between interactions appears to signal lack of observation rather than truly missing protein-protein interactions, suggesting that OGs should retain interactions across species even if their interactions have been observed in just one species.

Certain functional groups contribute more extensively to the meta-interactome than others, potentially reflecting the occurrence of more common types of protein-protein interactions across bacteria in general. In **Fig. 3-G**, I determined the overrepresentation of functional crosstalk between orthologous groups based on the underlying interactions between different proteins in the consensus meta-interactome. Unsurprisingly, the dominant category of interactor functions is “poorly characterized” (category S). Proteins with roles in translation or translational regulation (category J) also figure prominently, especially in interactions with poorly characterized groups.

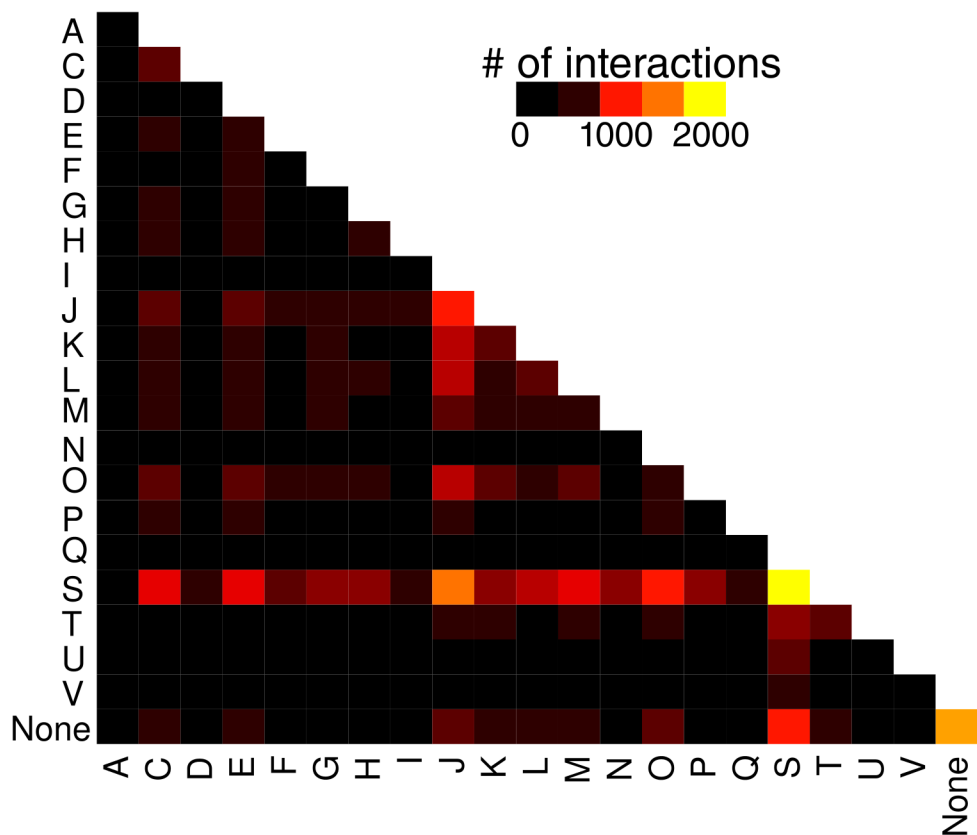


Fig. 3-G. Cross-functional interactions in the consensus meta-interactome. Here, *raw counts* of interactions within the consensus meta-interactome are categorized based on the functional category of each interactor, highlighting the incidence of cross-functional activity. See Methods for full key to functional categories. While most interactions appear between the same functional classes, I also observe that most functional cross-talk originates from OGs with translational or translation-related functions (category J) and with poorly characterized proteins (category S).

3.4.3 The meta-interactome predicts interactomes and their size

The construction of a meta-interactome as described above can be used to predict the interactome of any species with or without interaction data. I used the consensus meta-interactome as a model to predict any potential interactions in a given proteome independently of the availability of protein interactions in the underlying organism. In particular, I considered all interactions between OGs of a given proteome of an underlying organism. As such, I consider all proteins of the given proteome as interactors if I find their corresponding OGs interacting. As such, the interactome of a well-studied species such as *E. coli* can be improved by predicting yet undetected PPIs using data from a related but distinct species.

This simple prediction method was used with all protein-coding genes from each of several representative bacterial species of varied genome and proteome size (**Table 3-E**). Out of all eleven bacterial species shown, six have had comprehensive protein interactomes published, and the data is reflected in the total number of proteins participating in PPI with experimental evidence. To obtain a starting point for predictions, I used the interactome size estimation methods developed by Stumpf et al. (2008). These methods primarily depend upon the number of interactors and interactions in an experimental interactome to predict the true interactome size and therefore account for interactions not detected in the interactome (see **Fig 1-A** for the conceptual example). Here, I used the Stumpf et al. methods with three different counts of interactors and interactions: those from each of the six published interactomes, the larger counts found

in the meta-interactome, and the fraction of the interactome derived from experimental data. In cases where a given species has been the subject of just one comprehensive interactome study (e.g., with *Synechocystis*), the counts provided by the first option are very similar to the third.

Table 3-E. Predicted bacterial interactome sizes.

Species and Strain Name	Predicted interactome size in PPIs (this study)	Predicted interactome* size, from published interactome, in PPIs ¹	Predicted interactome size, from meta-interactome, in PPIs ¹	Predicted interactome size, from meta-interactome (experimental PPI only), in PPIs ¹	Proteins in proteome (vs. proteins in meta-interactome)
<i>Bacillus subtilis</i> str. 168	17146	N/A	117229	67921	4175 (1597)
<i>Caulobacter crescentus</i> CB15	25792	N/A	177318	1580788	3885 (1482)
<i>Escherichia coli</i> K-12	43702	25736	62770	30087	4306 (3593)
<i>Helicobacter pylori</i> 26695	10576	13275	14271	5455	1553 (1337)
<i>Mesorhizobium loti</i> MAFF303099	57905	50735	256414	50838	7272 (3456)
<i>Mycoplasma genitalium</i> G37	718	N/A	7331	N/A	475 (149)
<i>Pseudomonas aeruginosa</i> PAO1	47815	N/A	88143	7073281	5892 (2488)
<i>Salmonella enterica</i> subsp. <i>enterica</i> serovar Typhi	30788	N/A	268219	554147	4607 (2723)
<i>Staphylococcus aureus</i> NCTC 8325	9339	N/A	59233	650299	2767 (1099)
<i>Synechocystis</i> sp. PCC 6803	27816	11221	66575	11811	3575 (2311)
<i>Treponema pallidum</i> str. Nichols	6722	7433	10350	7762	1036 (835)

¹ As per method of Stumpf et al. (2008).

* Published interactomes are those specified in **Table 3-C**.

The interactome size prediction methods in this study are the results of predicting that two different orthologous group members will interact as long as members of the two groups have been observed interacting in any bacterial species. The resulting totals are shown in the second column (Predicted interactome size from meta-interactome (this study)). Results from the interactome size prediction method

used by Stumpf et al. (2008) are shown here for comparison: where possible, these are used with interaction and interactor totals from published interactomes (third column). Two hybrid approaches are also presented, with the input for the Stumpf method provided by the total counts of interactors and interactions predicted by the interactome (fourth column) or by the experimentally-observed interactions in the meta-interactome only (fifth column). The final column provides the count of proteins in each respective proteome along with the fraction of those proteins present in the meta-interactome, including all proteins involved in functional predictions.

Considering a set of reference proteomes (**Table 3-C**), I found that interactome sizes thus obtained appear to increase linearly with the proteome size of the underlying bacterial species (**Fig. 3-H**). For example, the *E. coli* genome codes for more than 4,000 unique proteins, and more than 3,000 of which have been found to participate in at least one PPI in one or more studies. The *B. subtilis* genome codes for roughly the same number of unique proteins but fewer than 1,000 of these proteins have been found to participate in PPIs. However, *B. subtilis* has also been studied much less extensively, hence these numbers do not reflect the true number of interactions in a cell.

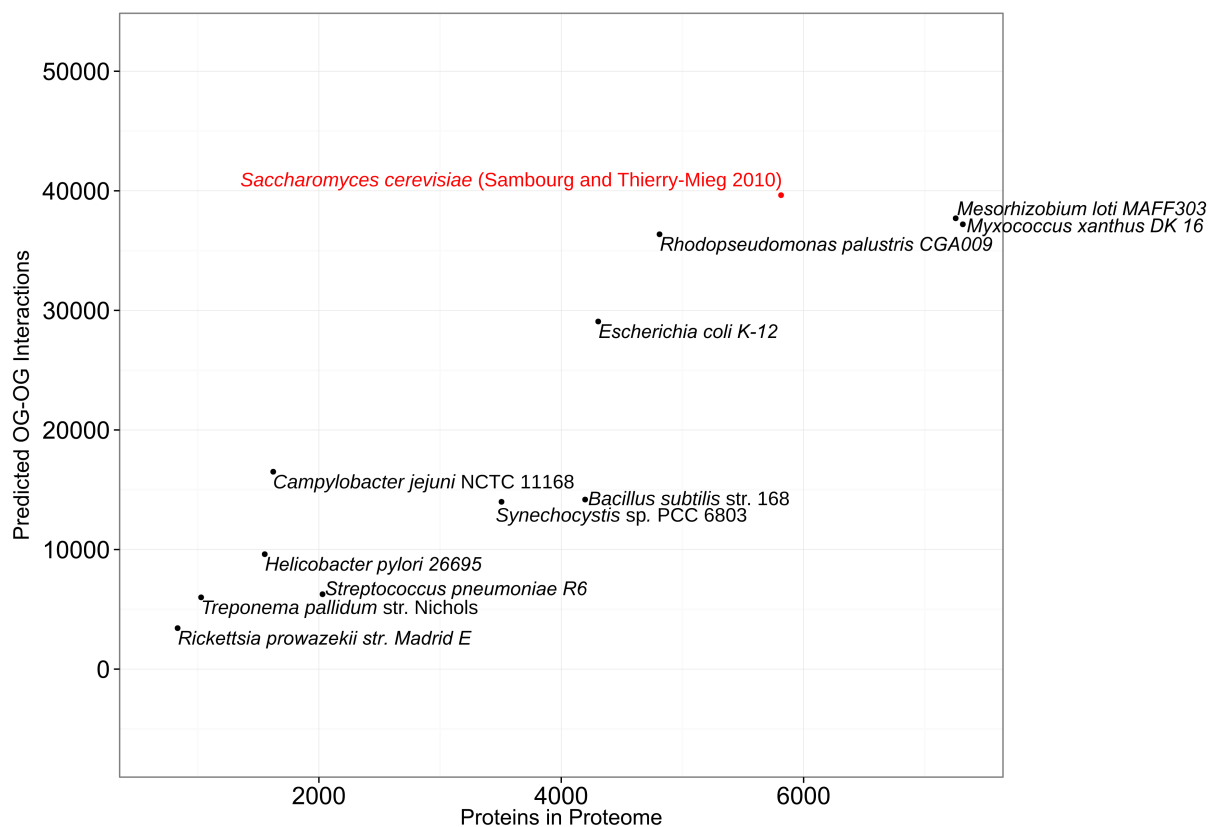


Fig. 3-H. Predictions of maximal interactome size. Based on the consensus meta-interactome, I show the upper bounds of predicted interactome size (in number of protein-protein interactions, or PPI) as a function of proteome size. Each point corresponds to the Uniprot reference proteome of a single species and strain. The *Saccharomyces cerevisiae* interactome size predicted by Sambourg and Thierry-Mieg (2010) is provided for comparison and shown in red.

The more interactions are detected, the fewer are left to be predicted. As a result, unstudied or incomplete interactomes have the largest potential for prediction. For instance, there are very few PPIs known from *Streptococcus pneumoniae*: just 63 of the 2,030 proteins coded for in the *S. pneumoniae* R6 genome have experimental interactions in IntAct. My predicted interactome for this protein set increases that total to 850 proteins. Similar results are seen for *B. subtilis* and for *Mycobacterium tuberculosis*.

3.4.4 Biological differences vs. technical differences in interactomes

Published interactomes vary in size and composition across different studies and species, rendering them difficult to compare. In the case of *Campylobacter jejuni*, a genome of 1,654 ORFs yielded an interactome of more than 11,000 distinct PPIs from yeast two hybrid (Y2H) screens using ~90% of the ORFs, or 1,477 in total (Parrish et al. 2007). By contrast, the interactome of *M. loti* as reported by Shimoda et al. (2008) includes just over 3,100 PPI though its proteome contains 7,281 predicted proteins. These discrepancies are clearly determined by different coverage: in the case of the *M. loti* interactome, the full genome was used as yeast two hybrid preys but only 1,542 of 7,281 genes were used as baits. This subset was selected as per the goals of the study and therefore represents a conscious technical difference between interactomes.

The comparison of interactomes also reveals unavoidable methodological discrepancies. More than half of the PPIs contributing to the meta-interactome were observed using two hybrid methods, offering some methodological consistency, yet these methods may vary in technical implementation details such as protein expression

conditions, growth conditions, or even the exact yeast or bacterial strains used. As I have shown previously, even when exactly the same protein pairs are tested by Y2H assays, small differences in the experimental protocol can yield dramatically different results (Chen et al. 2010; Caufield et al. 2012). Inclusion of affinity purification and mass spectrometry (AP/MS) approaches introduces another concern: AP/MS methods typically infer interactions from co-purification through a spoke model approach (that is, that a single bait is assumed to interact with all of its co-purified proteins) (Abu-Farha et al. 2008) while two hybrid methods generally screen for binary interactions only. Previous work has estimated that the spoke-model approach over-estimates the number of PPIs by about 3-fold (Rajagopala et al. 2014).

In this study, I have attempted to reduce the impact of technical differences between interaction studies by focusing on the subset of interactions observed in multiple species. This approach is especially effective for minimizing the influence of potentially erroneous spoke model interactions, as the bulk of these interactions in the meta-interactome are from just two species (*E. coli* and *M. pneumoniae*, both of which have been subjects of full protein complex surveys). Even after interaction screens are completed, however, filtering conditions determine the line between raw and final data and hence the data deposited in databases such as IntAct. With less frequently studied bacterial species like *Synechocystis* (Sato et al. 2007), researchers may observe too many PPIs among proteins of unknown function to accurately determine false positive interactions. It is crucial that all experimental details are documented with as much detail as possible, not only to ensure reproducibility, but to improve our ability to

understand differences between species. In the meantime, I believe a cross-species approach is helpful for identifying expected PPI in interactomes.

The cross-species approach employed here takes advantage of similarities in protein interactor sequence and similarities in interaction data to account for the impact of technical differences. As seen in **Fig. 3-F**, fewer than one thousand OG vs. OG interactions in the meta-interactome have been observed in more than one bacterial species, yet more interactions should be conserved across any two pairs of bacterial species.

Finally, some differences among interactomes may be due to real distinctions in genetics and physiology. Many processes show considerable genetic variation in bacteria, even when they are traditionally considered to be highly conserved. For instance, ribosomes are surprisingly malleable (Shoji et al. 2011, Wilson and Nierhaus 2005) as are flagella (Titz et al. 2008), cell division proteins (Margolin 2009) or protein complexes in general (Caufield et al 2015, see also Chapter 2 of this work). A more complete meta-interactome should therefore shed light on the biological differences between species.

3.4.5 Meta-interactomes reveal broadly-conserved interactions involving proteins of unknown function

Of all OG-OG interactions involving OGs of unknown or unclear function (UF OGs), fewer than 10 are seen in more than 2 different species (**Fig. 3-F**). Highly conserved PPIs are thought to serve more fundamental processes in a cell (e.g. Häuser et al.

2012, Rajagopala et al. 2014), hence I identified well-conserved interactions for function prediction. Some of the most frequently observed PPIs (specifically, OG-OG interactions) across species are interactions among enzyme subunits, e.g. the alpha and beta subunits of tryptophan synthase (**Table 3-F**).

Table 3-F. Conserved interactions involving selected OGs of unclear function.

Interactor A	Interactor B	Functional Category and Function (A)	Functional Category and Function (B)	Species
ENOG4105W16	ENOG4105W16	S - Blue light sensor protein	S - Blue light sensor protein	<i>Synechocystis</i> sp. PCC 6803, <i>Thermosynechococcus elongatus</i>
ENOG4105CXV	ENOG4108XPN	S - Gliding motility protein	S - Roadblock Ic7 family protein	<i>Thermus thermophilus</i> , <i>Myxococcus xanthus</i>
ENOG4108WXF	ENOG4108WXF	S - KaiA , Component of the KaiABC clock protein complex	S - KaiA , Component of the KaiABC clock protein complex	<i>Thermosynechococcus elongatus</i> , <i>Synechococcus elongatus</i>
ENOG4105K7D	ENOG4108UKE	S - Ribosome maturation factor RimP	J - 30S ribosomal protein S12	<i>Campylobacter jejuni</i> , <i>Helicobacter pylori</i>
ENOG4105ZRE	ENOG4108YZA	S - Protein of unknown function (DUF3539)	E - GlnB, Nitrogen regulatory protein P-II	<i>Nostoc</i> sp. PCC 7120, <i>Synechococcus elongatus</i>
ENOG4105QDU	ENOG4108V9G	S - Uncharacterized protein	S - Uncharacterized protein	<i>Campylobacter jejuni</i> , <i>Helicobacter pylori</i>
ENOG4108SDW	ENOG4107QMP	S - recombination protein RecO	L - DNA polymerase III gamma and tau subunits	<i>Campylobacter jejuni</i> , <i>Helicobacter pylori</i>

All interactions in this table have been observed in at least 4 PPI across bacterial species of at least two different genera, with species identified in the Species column. The full list of interactions in this set is provided in **Appendix Table IV-D**.

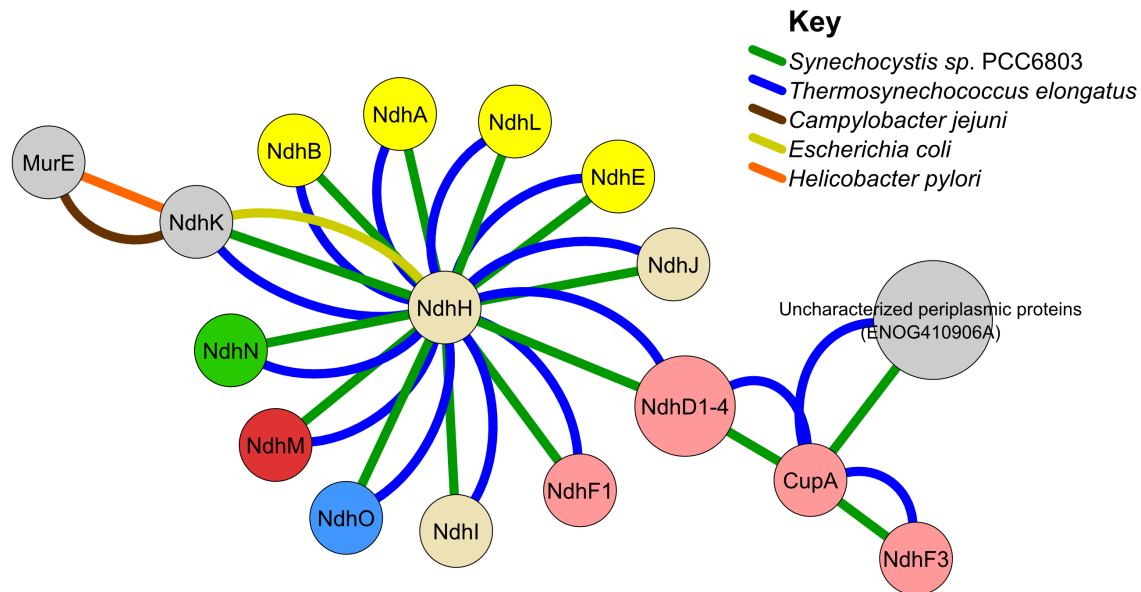
This list omits broadly-conserved self-interactions, such as those among histidine kinases (ENOG4105BZU). An orthology-based approach is more informative when used

with interactions among proteins in different groups (in this case, different OGs) than with interactions among proteins of the same OG as individual protein identities are ignored in the consensus meta-interactome. I have made the assumption that cross-OG interactions are more likely to indicate cross-function interactions and are therefore of great relevance to functional context.

MdaB (ENOG4105NF4) proteins figure prominently in the meta-interactome. MdaB was first identified as modulator of drug activity (Chatterjee et al. 1995) and is still annotated as such in most databases. Later, Wang et al. (2004) characterized it as a novel antioxidant protein similar to NADPH nitroreductases which play an important role in managing oxidative stress essential for successful colonization of *H. pylori* in its host (Wang et al. 2004). Its mutants are unable to colonize human host cells (Wang et al. 2004). However, the MdaB interaction network indicates another unrelated function as it interacts with three motility related proteins in three different species: a chemotaxis protein (UniprotKB: O25152) from *H. pylori*, flagellin C (UniprotKB: P96747) from *C. jejuni*, and chemotaxis protein CheW (UniprotKB: P0A964) from *E. coli* K-12. The colonization phenotype may be related to motility rather than oxidative stress. In fact, motility is critical for initial colonization of *H. pylori* in its host cells (Ottemann and Lowenthal 2002). FlaC in particular is well characterized as an important factor for host cell invasion in *C. jejuni* (Song et al. 2004).

Interactions between components of a protein complex can be reconstructed from the meta-interactome interactions. The cyanobacterial NDH-1 membrane protein complexes provide a good example: these proteins belong to widely-conserved family of energy converting NAD(P)H: Quinone oxidoreductases which are unique to organisms capable of photosynthesis. Many distinct NDH-1 complexes may coexist in cyanobacteria to carry out different functions like respiration, cyclic electron transfer and CO₂ uptake (Zhang et al. 2005, Battchikova et al. 2011, Korste et al. 2015). At least four NDH-1 complexes are predicted in cyanobacteria in *Synechocystis* 6803 (L, L', MS, MS'). Each complex is composed of a basal complex (NdhA-C, NdhE,G-K, NdhL-O) associated with variable subcomplexes of Ndh and Cup subunits (**Fig. 3-I-A, B**). Each complex has a different function: for example, NDH-1L and L' are responsible for respiration and cyclic electron flow and NDH-1MS/MS' for CO₂ uptake. The multitude of functionality of cyanobacteria is possible due to the presence of a great diversity of ndhD (D1-D6) and ndhF (F1, F3 and F4) gene families. It is possible that with sudden changes in CO₂ levels, cyanobacteria can flexibly use the NDH-1M basal subcomplex and change contents of its variable subcomplex to form MS and L complexes (Korste et al. 2015).

A



B

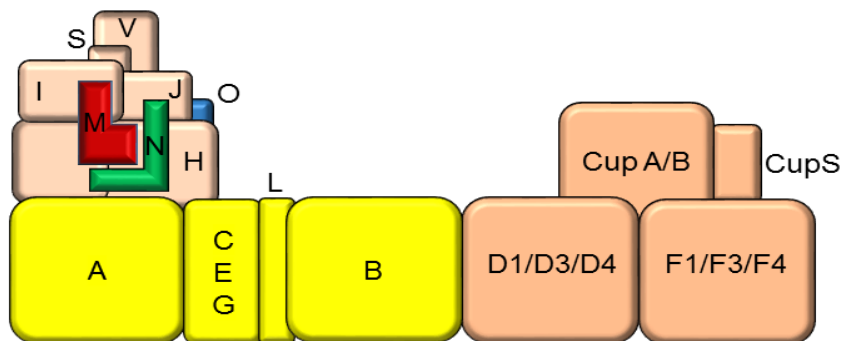


Fig. 3-I. The NDH-1 complex as an example of conserved interactions. (A) A NDH component interaction network from multiple species. Each node in this network corresponds to a single orthologous group and is labeled with its eggNOG identifier and most common protein name(s). Groups colored in orange are known components of the *E. coli* NDH-1 complex, the CupA protein group is shown in gray, and likely accessory proteins are colored in blue. Interactions between any proteins in two groups are shown as edges; edge width corresponds to the total count of protein interactions in the meta-interactome. Except where noted, all interactions in this network were experimentally observed with proteins from *Synechocystis* sp. PCC6803 and from *Thermosynechococcus elongatus*. (B) A model of the NDH-1MS complex in cyanobacteria. Each box corresponds to a protein or group of proteins; those labeled with a single letter are Ndh proteins. Figure adapted from He et al. (2016).

An example of the NDH-1MS (NDH-1M, NdhD/F/CupA/CupS) network in *Synechocystis* 6803 and *T. elongatus* BP-1 is shown in **Fig. 3-I-A**. Only one similar interaction (NuoD and NuoB) is observed in *E. coli*. CupA (ENOG4107YAI) has been found to interact with NdhF (ENOG4106TXZ), NdhD1-D4 (ENOG4105C8S), and an unknown protein (ENOG410906A) to form the NDH-IS (NdhD/F/CupA/CupS) sub-complex. ENOG410906A, though a protein of unknown function, has sequence similarity to Fasciclin superfamily proteins associated with cell adhesion in plants and algae species. This protein is 133 amino acids with a predicted molecular weight of 13 kDa. Korste et al (2015) found a similar protein (UniprotKB: P73392) in *Synechocystis* 6803 and Q8DMA1 in *T. elongatus* BP-1 and designated it as CupS, a small subunit of the NDH-1MS complex. The NMR studies, showed that though the protein was structurally similar to Fasciclin superfamily, but was not associated with adhesion, contrary to Fasciclin superfamily proteins due to its intracellular location. Though, CupS has been shown to interact with NdhD/NdhF/CupA, its function is still unknown. This network data not only provides clarity about the interaction of NDH-1 complex proteins but also predicts a probable function of this unknown protein ENOG410906A in respiration. Cyanobacterial meta-interactome networks (**Fig. 3-I-A**) clearly show that the NdhH subunit interacts directly with all associated subunits, a point which had been missing in all predicted structures of NDH-I.

3.4.6 Interactomes are impacted by high-throughput experimental methods

The data in the meta-interactome is the result of a variety of methods and therefore is directly influenced by variations in the methods. As shown in **Fig. 3-J**, the majority of the PPI contributing to this data set are the result of two hybrid screens. It is therefore valuable to ascertain how much of the meta-interactome content is potentially the result of methodological variation.

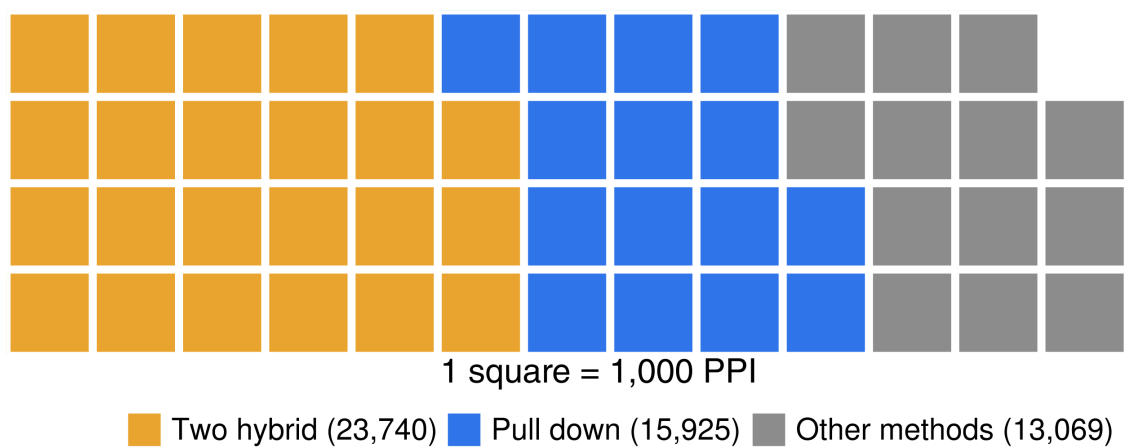


Fig. 3-J. Composition of the meta-interactome by interaction detection method. Each box represents 1,000 protein-protein interactions (PPI). Counts in parentheses are total PPI. *Two hybrid* includes interactions annotated as either “two hybrid” or “two hybrid array”.

Here, I use a clustering approach to compare all methods that have been applied to the Braun et al. (2009) gold-standard dataset (**Fig. 3-K**). Two methods may detect similar interactions yet these methods detect different subsets of the total set of all possible interactions. Clustering provides the benefit of going beyond sums of interaction results in that it compares the patterns of results, revealing further differences between experimental methods. There is one distinct caveat regarding these methods: the Braun et al. positive reference set includes only human proteins rather than bacterial proteins. Some methods presented here, such as MAPPIT (Tavernier et al. 2002), are designed specifically to detect mammalian PPIs. It is possible that distinctions between bacterial proteins may create even more complications with regard to methodological differences as their expression conditions may differ more from nature (e.g., with bacterial proteins expressed in yeast vs. human proteins expressed under the same conditions).

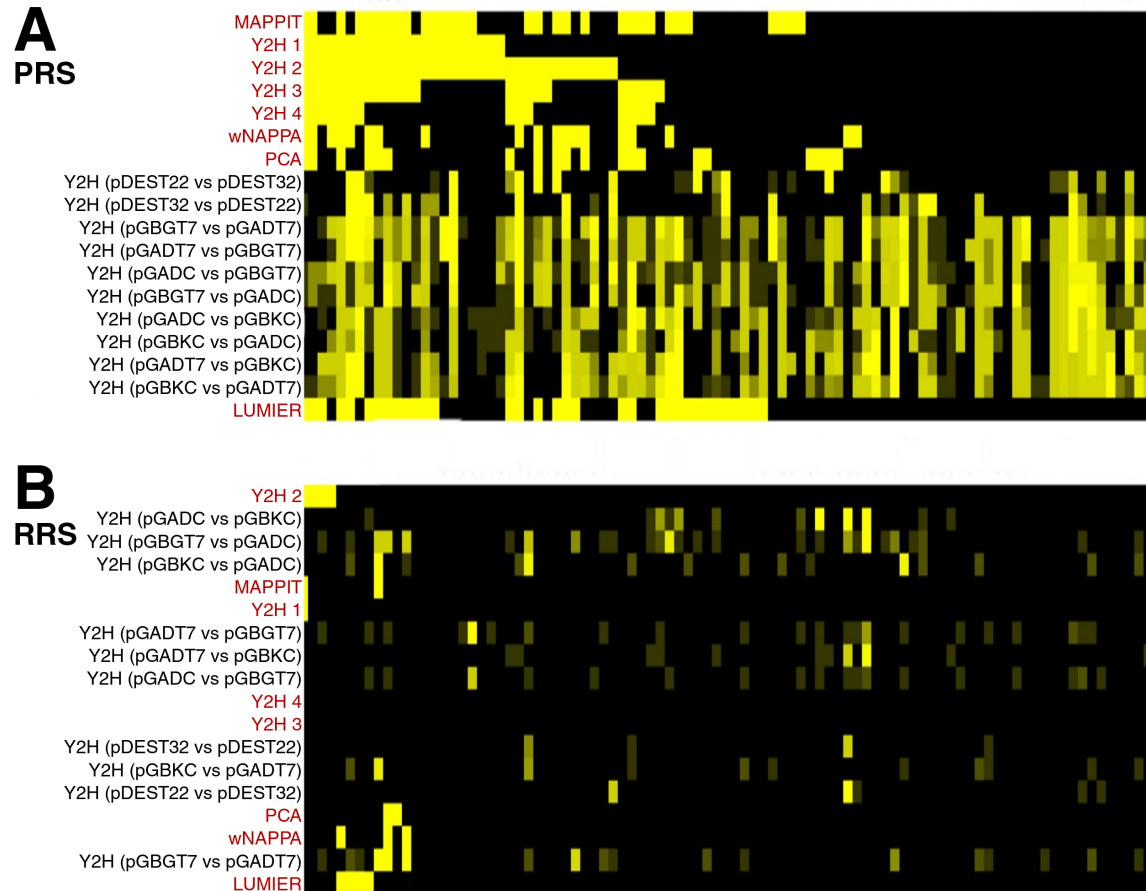


Fig. 3-K. A comparison of high-throughput yeast two hybrid screens. The assays used to detect each protein pair in the **(A)** positive reference set (PRS) and **(B)** random reference set (RRS), as defined by Braun et al. (2009), are clustered by the number and similarity of the interactions detected across the respective reference set. For Braun et al. assays (indicated by red labels at left), columns denote whether a specific protein interaction was reported. For all other assays, values are weighted values as described in the Methods, with increasing brightness indicating greater value. Black spaces indicate that no interaction was detected. Reference set numbers and exact methods have been omitted for clarity; see Caufield et al. (2012) for further details. Figure adapted from Caufield et al. (2012).

When the available positive reference set data is reduced to binary decisions regarding whether an interaction is visible (i.e. without considering 3-AT concentrations), the results are striking: the Y2H results and the Braun results each cluster together very consistently except for the pDEST vectors (which were also used by Braun et al.). Not surprisingly, the Braun Y2H assays with the same vectors – but different reporters – clustered together, with the two-reporter assays simply producing fewer interactions (Braun et al. 2009). The Y2H assays were notable as bait/prey swaps (that is, switching the vectors used to express bait and prey protein fusions) typically clustered together too, e.g. in the pDEST, pGBGT7–pGADT7, and pGBGT7–pGADC assays, but not in the pGBKC–pGADC/pGADT7 cases (**Fig. 3-K**). This is surprising, as even bait/prey swaps usually result in distinctly different interaction patterns, and this result is not immediately obvious when examining the raw data. Overall, these results indicate that each method may detect different subsets of interactions within the same set of protein pairs, especially when multiple sets of Y2H results are compared.

These comparisons hold implications for the total number of interactions in the meta-interactome. Out of the 23,740 PPIs in the set identified using yeast two hybrid methods in particular, if I assume that at least 10% of biologically-relevant PPIs are not captured by these methods in some way (as indicated by the inability of even the least stringent of some yeast two hybrid methods to detect members of the PRS; **Fig. 3-K-A**), then the total number of PPIs in the meta-interactome could increase by more than 2,000 interactions. If, out of a set of 100 potential PPI, some methods may only detect 25 to 30 interactions, then there are likely numerous additional PPIs existing in nature but not covered by the meta-interactome. Starting with about 13,000 PPIs not identified through

yeast two hybrid or pull down methods, these missing counts may include as many as 9,750 additional PPI among all bacterial interactors.

3.4.7 Further discussion

If I assume that the average degree of a protein remains the same, independent of the proteome, then interactomes should grow linearly with proteome size and thus with genome size (**Fig. 3-H**). However, bias in the available data is likely creating distorted predictions: the *E. coli* data point (at the top of the figure) does not fit the trend and most of the PPI predictions I can make originate with *E. coli* data. Additionally, predicted interactome sizes are limited by the number of unannotated or highly unusual genes in a genome. In the case of the largest genomes in this set (*P. aeruginosa* and *M. loti* have genomes larger than 6 Mb), both contain at least 300 genes without orthology predictions (*P. aeruginosa* contains ~310 while *M. loti* contains 737). Further annotation of these genes or interactions among their products may allow for interaction predictions more like those for other species.

Proteome size is likely just one trait contributing to the overall complexity of a species (Schad, Tompa, and Hegyi 2009) and the interactome of that species may represent just one facet of its complexity. Some methods used to estimate interactome size were intended for use with human or yeast proteins (Venkatesan et al. 2009, Sambourg and Thierry-Mieg 2010) and should likely work for bacteria but may abstract or even ignore bacteria-specific physiological phenomena in the process. The methods employed by Venkatesan et al., for example, rely upon estimates of the false positive rate when screening human proteins for interactions; this rate is inconsistent between methods, species, and even subsets of proteins from a given species. Another confounding factor is that false positives are likely to grow exponentially with increasing proteome size, e.g. because a fraction of proteins interact non-specifically with hydrophobic surfaces.

In the cases of some interactions with proteins of unknown function, the lack of functional characterization may be a lack of understanding both proteins' role in a larger complex. In these cases, better functional characterization may require an understanding of the complex's biological role. From the reverse perspective, however, if these protein components are involved in PPI outside of their usual complex, the additional interactions may reveal properties shared by other complex components. Some commonly observed interactions involve poorly-understood groups of proteins: proteins in NOG12793 are known to be calcium ion binding (56,354 proteins in 1,123 species map to this OG) but no other property can be defined for the entire group though it consistently includes hyalin domains (these domains are generally associated with cellular adhesion, as per Callebaut et al. 2000, though they may have a variety of roles).

The meta-interactome approach is an intentional abstraction. It is intended to underscore the bacterial cross-species commonality and conservation of protein interactions among currently available interaction data. As a result, this approach is limited by at least three main factors: limitations of protein-protein interaction screens, limitations of publicly-available data, and constraints on orthology prediction. All experimental interactomes are inherently incomplete and may include numerous false positives and otherwise erroneous results. The authors of these studies employ different filtering approaches and likely interpret their results based on expectations (e.g., some interactome studies eliminate frequently-interacting proteins like chaperones from their screens). Most of the available interaction data for bacterial proteins has focused on just a handful of species. Additional screens of proteins from more diverse sources across

the bacterial tree of life will reveal a universe of yet unknown functions, just as gene sequences did for genetic diversity.

Chapter 4 – Assessing Bacterial Protein Function using Bacteriophage Proteins

Portions of this chapter have been published as:

Mehla, J., Dedrick, R.M., Caufield, J.H., Siefring, R., Mair, M., Johnson, A., Hatfull, G. F., Uetz, P. (2015). The protein interactome of mycobacteriophage Giles predicts functions for unknown proteins. *J Bacteriol*, 197(15), 2508–2516.
doi:10.1128/JB.00164-15.

4.1 Abstract

Bacteriophage infections are likely the most common type of biological interaction on Earth. Any comprehensive study of microbial evolution must therefore consider interactions between bacteriophages and their bacterial hosts. Here, I have assembled and curated a set of experimentally-verified interactions between proteins from 10 bacterial species and 29 bacteriophages, sourced from 48 different studies. Unlike previous studies of phage vs. host interactions, I have specifically focused on direct protein-protein interactions rather than indirect mutational comparisons or predicted impacts on phage infectivity. I have used this new resource to further analyze the most frequently observed types of protein interactions between bacteria and their viruses. These resources will ideally serve as a basis for further study into bacteria vs. bacteriophage interactions, bacterial protein function, and potential targets for antibacterial or bacteriostatic phage therapy.

4.2 Introduction

4.2.1 Microbiology in the context of bacteriophage interactions

The bacterial occupants of a microbiome exist alongside those of a virome, or the set of all viruses in the biological niche. The viruses infecting bacteria in particular are collectively referred to as bacteriophages. Considered at a global scale, the number of bacteriophages present on Earth is staggering: starting with a rough population estimate of 10^{30} bacterial cells on the planet – mostly in the oceans – various estimates have suggested as much as a 100-fold greater population of bacteriophages (Wommack and Colwell 2000, Rowher 2003, Clokie et al. 2011).

These viruses serve as a massive and constant source of new genetic variation, both as the result of phage-mediated genetic transfer (that is, transduction) and through the perpetual battle between the viruses and their hosts. Bacteriophage must constantly develop new methods to infect, control, and eventually lyse bacteria, while bacteria must defend against these lethal results through systems like CRISPR (Barrangou et al. 2007) or quorum-sensing dependent defenses (Høyland-Kroghsbo et al. 2013).

Bacteriophages have served as ideal models for much of the history of molecular biology, but unfortunately this favored status has not smoothly translated into the genomic age. For decades, researchers studying phages used mutational analysis-based techniques they knew would get results. They lacked easy methods to observe direct, protein vs. protein interactions and focused on more easily observable

interactions at the membrane surface (i.e., the interactions most crucial to a phage's infection of its host). Many current studies of phage biology continue to use similar methods: work by Washizaki et al. (2016) on bacteriophage T4 tail fibers primarily employed phage mutants and plaque assays, for example. The consistent use of reductionist methods focused on easily observable phenomena (i.e., numbers of infected bacteria and of replicating phages) lends noticeable consistency to the field but fails to address the potential secondary effects of mutations. In a potentially counter-intuitive way, in order for the field of phage-host interactions to adapt to a systems biology perspective, it must focus more closely on the interactions between individual proteins rather than disruption of entire systems at a time.

4.2.2 Extending interaction analysis to viral proteins

Protein-protein interactions between phage and their hosts are, with a few exceptions (Roucourt and Lavigne 2009, Blasche et al. 2013), largely unexplored. Some researchers have developed databases specifically for the curation of virus vs. host protein interactions: VirHostNet (Guirimand et al. 2014) and VirusMentha (Calderone et al. 2015). Of these two databases, VirHostNet contains just two phage vs. host interactions (involving Myoviridae, specifically). Due to the intimate relationships between bacteria and viruses, I chose to use these interactions as a novel venue for exploring bacterial genes of unknown function.

I elected to explore the current state of the field of phage-host protein interactions for two primary reasons. First, while phage biology remains an active area of research, its findings are not always clear in the broader context of microbiology. This is potentially the result of researchers not wishing to overstate their findings. Each interaction identified between a host and viral protein may offer new insights into this common type of biological relationship, especially when considering the vast assemblage of gene sequences observed only in bacteriophage genomes. This variety is my second motivation for focusing on previously-studied phage-host interactions: even the genomes of well-studied bacteriophages contain sequences without known functions. I therefore hypothesized that an integrative approach to phage-host interaction data could reveal overall patterns relevant to phage-host relations.

4.3 Experimental methods

4.3.1 Data curation and data set assembly

Sets of interactions between proteins derived from bacteriophages (that is, having a sequence identical to that of a translated bacteriophage gene) and those from bacteria were curated from the literature over a period between March 2015 and September 2016. Literature was first selected from papers cited by Häuser et al. (2012) in their review of bacteriophage protein interactions and supplemented with works associated with interactions present in the IntAct (Kerrien et al. 2012), DIP (Xenarios et al. 2002), MINT (Licata et al. 2011) and BIND (Alfarano et al. 2005) databases. (The majority of the interactions in these databases is now available through IntAct.) The VirusMentha

project (Calderone et al. 2015) was also used to identify publications with a virus vs. host focus. Interactions were extracted from databases, supplementary materials, or directly from text. An interaction was narrowly defined as a direct interaction between two proteins, though this interaction may have been observed through an indirect phenotype change (e.g., two hybrid methods), affinity purification-based methods, or co-crystallization. This definition distinguishes the records in this data set from indirect interactions, e.g. those inferred from mutational analysis or phage plaque screens.

See **Table IV-A** in the **Appendix** for the full data set. The data set contains the following values, with each line referring to a single interaction reported by a single study (for this reason, an interaction may appear multiple times in the set if it has been observed in multiple studies).

Phage_Interactor	The protein name of the phage interactor.
Phage	The name of the bacteriophage source of the phage interactor.
Phage_UPID	The Uniprot entry ID of the phage interactor.
Phage_Alt_ID	An alternate ID for the phage interactor if a Uniprot ID is not available. Otherwise, this is identical to Phage_UPID.
Phage_OG	An eggNOG v.4.5 orthologous group assignment for the phage interactor, if available. Otherwise, this is identical to Phage_Alt_ID.
Host_Interactor	The protein name of the host interactor.
Host	The species name of the bacterial host and source of the host interactor. Different strain identities are ignored in this data set.

Host_UPID	The Uniprot entry ID of the host interactor.
Host_Alt_ID	An alternate ID for the host interactor if a Uniprot ID is not available. Otherwise, this is identical to Host_UPID.
Host_OG	An eggNOG v.4.5 orthologous group assignment for the host interactor, if available. Otherwise, this is identical to Host_Alt_ID.
ExpMethod	The experimental method used to observe the interaction, as one of the methods defined by the PSI MI 2.5 methods ontology; see also Hermjakob et al. (2004).
ExpMethodID	The ontology ID for the experimental method. Details about all ontologies may be found through http://www.ebi.ac.uk/ols/ontologies/mi .
InfMethod	"Spoke" if the reported interaction is the product of a spoke expansion model. "-" if otherwise.
Source	The first author and publication year of the source of the reported interaction.
SourceID	An NCBI PubMed ID for the source.
Database	The source database, if present in a database of protein interactions, or a review article including a collection of interactions.
Incidence_Phage_OG	Total count of times the OG corresponding to the phage protein interactor is present in this table.
Incidence_Host_OG	Total count of times the OG corresponding to the host protein interactor is present in this table.

4.3.2 Mycobacteriophage Giles protein-protein interactome

Methods in this section refer to those performed by Mehla et al. (2015). Briefly, 74 of 77 ORFs in the genome of Mycobacteriophage Giles were cloned into each of four vectors and screened in high throughput using the yeast two hybrid method. Each bait (DBD-X) was mated with each prey (AD-Y) on rich medium (YPD plus adenine) in a 384-colony format for 36 to 48 h at 30°C. Diploid cells were selected for by pinning cultures from mating plates onto selective agar plates (–Leu –Trp) and growing them for 2 to 3 days. The diploids were then screened for interacting pairs by pinning them onto selective screening medium (–Leu –Trp –His) and incubating at 30°C for another 4 to 7 days. All baits (including self-activating baits) were screened on –Leu –Trp –His plates containing 3-AT to suppress nonspecific background; at least two different 3-AT concentrations between 1 and 100 mM were used for each screen to avoid elimination of true positives. The plates were monitored each day and positive colonies were evaluated with respect to the background growth on each plate.

I filtered out nonspecific raw Y2H data on the basis of prey count, with a few exceptions. Prey count is defined as the number of times a defined prey protein is found to be an interacting partner for any other bait in the tested set. The preys found to interact with 12 or more baits (an arbitrarily defined value specific to the raw data set only) were predicted to be the result of nonspecific interactions and were, with some exceptions, not included in the retest Y2H data set. A sticky prey was included in the retest data set

if it was found to interact specifically and strongly at a 3-AT concentration with no background growth visible on the same plate.

I used the filtered set of raw protein-protein interactions to form a retest set. These interactions were tested as described above in a 384-colony format in quadruplicate (each colony was plated four times on each plate) for each bait-and-prey combination in all different vector configurations. Fresh bait-and-prey arrays were prepared specifically for these retests. All protein-protein interactions were quantitatively titrated against background using a series of different concentrations of 3-AT between 0 and 50 mM.

A score, % 3-ATS, was calculated for each interacting bait-prey pair using the formula $\% 3\text{-ATS} = (C_{\text{PPI}} - C_{\text{B}}/C_{\text{PPI}}) \times 100$, where % 3-ATS is the % 3-AT score calculated for each PPI, C_{PPI} is the highest concentration of 3-AT at which a PPI was scored, and C_{B} is the concentration of 3-AT at which background was observed. Thus, each interacting pair was assessed quantitatively and assigned a % 3-ATS which was used to calculate an overall interaction score (% IScore). Once all PPIs had been retested, the % IScore was used to select high-confidence PPIs. The % IScore was calculated as $\text{IScore} = 3\text{-ATS} + \sum w_k$, where 3-ATS is the 3-AT score assigned to each PPI as described above and $\sum w_k = w_1 + w_2 + w_3$, where w_1 is the weight value for PPIs detected in multiple vectors, directly proportional to the IScore ($w_1 = 0$ if a PPI was detected by only a single vector or 33 if detected by at least 2 vectors), w_2 is the weight value for reciprocal interactions, also directly proportional to the IScore ($w_2 = 0$ if not found in a reciprocal set of

interactions [e.g., A-B and B-A] or 50 if it is a reciprocal interaction), and w_3 is the weight value for the prey count, inversely proportional to the IScore ($w_3 = 0, -5, -10, -15, -20, -25$, or -30 for prey counts of 1, 2 to 5, 6 to 10, 11 to 15, 16 to 20, 21 to 25, or 26 to 30, respectively). Then, % IScore = (actual IScore for a given interacting pair/highest IScore observed for any interacting pair) $\times 100$.

Giles protein properties were investigated further by R. Dedrick. Each Giles protein was assigned an essentiality value based on that determined by Dedrick et al. (2013). All proteins determined to be likely essential for the phage lytic cycle, whether by experimental observation or by their role as phage structural components, were designated “essential.” All other gene products were designated “nonessential.” For mass spectrometry (MS) analysis, wild-type *Mycobacterium smegmatis* mc²155 was infected with mycobacteriophage Giles at a multiplicity of infection (MOI) of 3. At 30 min and 2.5 h postinfection, a 1-ml aliquot was centrifuged, the supernatant removed, and the cell pellet immediately frozen. A high-titer lysate of the bacteriophage cultured in *M. smegmatis* was cesium chloride band purified twice and then submitted for mass spectrometry (MS) analysis along with the samples from the 30-min and 2.5-h postinfection time points. The mass spectrometry was performed by the University of California at Davis Proteomics Core on an LC-MS/MS Q-Exactive as described by Pope et al. (2014). This study refers to three MS fractions: an early fraction (30 min postinfection), a late fraction (2.5 h postinfection), and the phage particle (whole virion only). Individual proteins may be present in more than one MS fraction.

4.3.3 Data analysis

Protein interactors were annotated in a semi-automated way with eggNOG v.4.5 orthologous groups. This version of the eggNOG database incorporates a partial list of virus-based OGs (Huerta-Cepas et al. 2015). In cases of unclear or unannotated orthology, the online sequence mapping tool and hidden Markov models provided by eggNOG were used to search the eggNOG Bacteria sequence database to find the best match for the protein sequence across all bacteria-level NOGs. In cases where proteins remained without a clear NOG match – primarily for bacteriophage proteins – individual BLAST (Altschul et al. 1997) *tblastn* searches (with default parameters, with the exception of a word size of 3 instead of 6) were performed for a subset of protein sequences to identify potential domain-level matches.

Network analysis was performed using Cytoscape v.3.4 (Shannon et al. 2003). Sets of protein complexes are identical to those used in Chapter 2 but filtered to include only protein components found to interact with bacteriophage proteins. Sequence alignments were prepared using Clustal Omega (Sievers et al. 2011) and visualized using Jalview (Waterhouse et al. 2009).

4.4 Results and discussion

4.4.1 An example of phage protein interactions from Mycobacteriophage Giles

As viruses, bacteriophages have life cycles which are inherently dependent upon interactions with their potential hosts. The same is true of the functions of the proteins involved in this process. Not every phage protein is directly involved in host interactions, however, as phages must also encode the components of their structure and perhaps even those providing functions we remain unaware of. From this perspective, it is therefore helpful to incorporate the interactions among bacteriophage proteins into the analysis of interactions between phage and host proteins.

Mycobacteriophage Giles provides a model for phage protein interactions and the potential for novel protein functions. Originally investigated by the Hatfull lab at the University of Pittsburgh, Giles is a phage known to infect *Mycobacterium smegmatis*, contains a genome of about 53 Kb and 77 predicted protein-coding genes, and is a genetic oddity among mycobacteriophages: more than half of the genes in the Giles genome appear to have little sequence similarity to any other mycobacteriophage genes (Morris et al. 2008). At least 35 of its genes are necessary for lytic growth (Dedrick et al. 2013). Giles therefore presented an excellent opportunity to discover functional roles of novel proteins necessary to a particular phage's life cycle. Using yeast two hybrid screening methods and mass spectrometry analysis, my lab defined a protein-protein interactome for this bacteriophage (Mehla et al. 2015; **Fig. 4-A**). As expected, some of the strongest reactions were those between the phage's structural components. Some proteins of unknown function, including Gp56, Gp57, and Gp60, strongly and consistently interacted with the predicted DNA methylase Gp62.

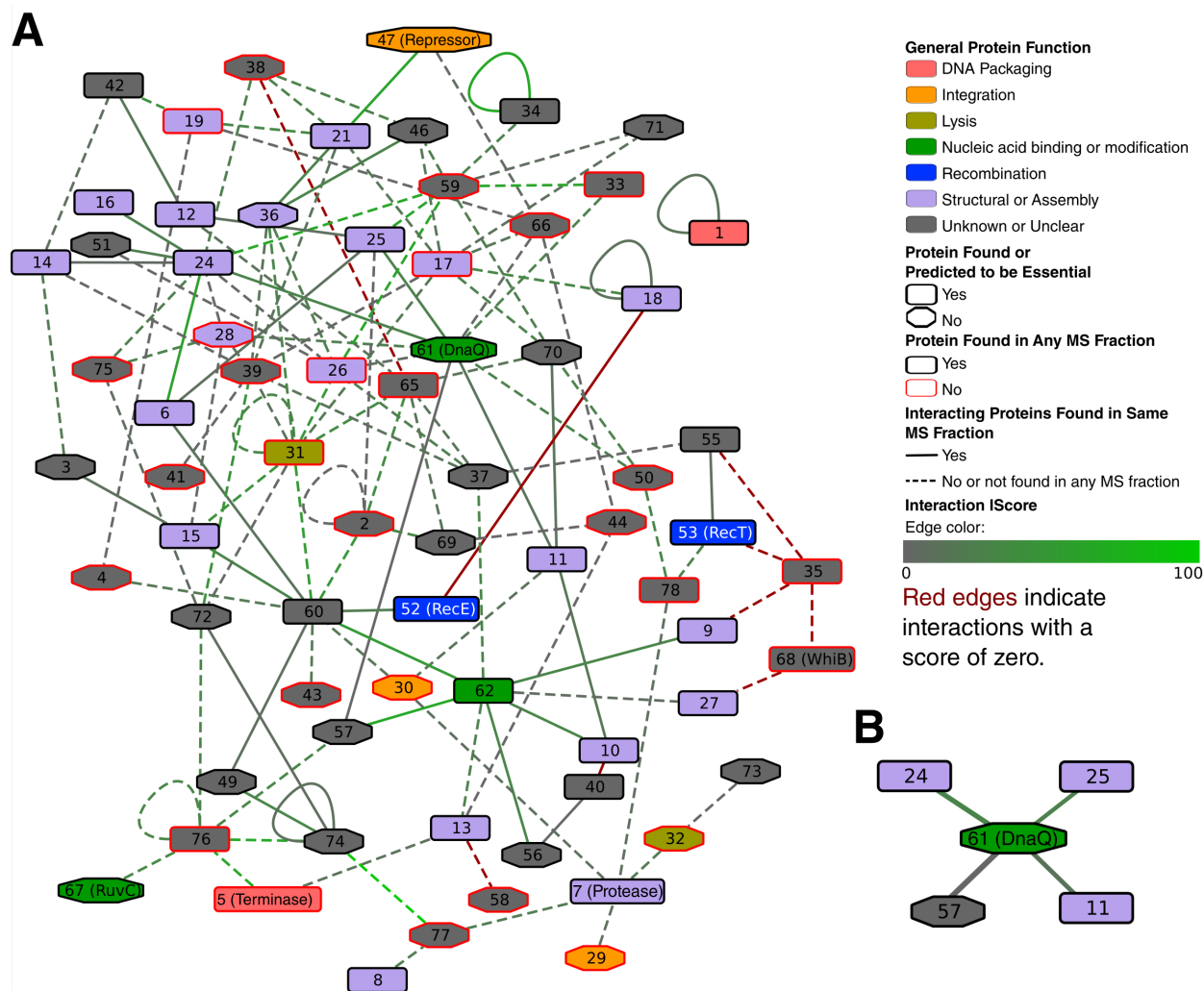


Fig. 4-A. The mycobacteriophage Giles interactome. A) Each node in this network refers to a specific gene product (Gp) coded for by the mycobacteriophage Giles genome. Interactions, shown as edges, are those identified by yeast two hybrid screens. Interaction IScore is a measurement of interaction strength and is described in the Methods. *Essential* refers to essentiality of the protein-coding gene to the Giles lytic cycle as determined by Dedrick et al. (2013). MS, mass spectrometry; *same fraction* refers to observation of the protein in one of three fractions as described in the Methods. Figure adapted from Mehla et al. (2015). **B)** Interactions involving Giles Gp61 (DnaQ) and proteins observed in the same MS fraction.

One interactor in particular, Gp61 (Uniprot: A8WA49), is a protein with sequence similarity to the *E. coli* DNA polymerase III epsilon subunit (DnaQ; Uniprot: P03007). Among all mycobacterial genomes, the Giles Gp61 sequence appears to be most similar to a sequence found in *M. canariasense*, though only a draft sequence of this species' genome has been published thus far (Katahira et al. 2016). It is therefore difficult to determine if the bacterial sequence is a host protein or a phage-derived sequence. In any context, the overall sequence similarities indicate the protein likely has an exonuclease functionality. Gp61 was found to strongly and consistently interact with several other Giles proteins (**Figure 4-A-B**), three of which have structural/assembly roles. These associations suggest several possibilities: Gp61 may also have a role in phage structure or assembly, the similarity to a bacterial protein may cause Gp61 interactors to interact with host proteins as well, and/or this protein interferes with host functions to mediate the phage life cycle.

At this point, I was left with two questions: what can a phage interactome like that of Giles tell us about phage vs. host interactions, and what can the other phage vs. host interactions in the curated set provide context for on their own? Out of all Giles proteins, we may expect those most likely to interact with host proteins are those with similarity to previously observed phage vs. host interactors and proteins *not* participating in strong phage vs. phage PPI. We would, however, expect potential phage vs. host interactors to be present in MS fractions isolated from an *in vivo* phage infection. As per the

interactome, the proteins fitting this criteria include Gp40, Gp53, Gp55, all proteins of unknown function, as well as the predicted head assembly protein Gp9.

Similarity between Giles protein sequences and that of other phages is surprisingly minimal. A sequence identical to that of Giles Gp40 appears in the related mycobacteriophages Evanescence, OBUPride, Kinbote, and HH92, but nowhere else apparent. There is some similarity between lambda NinD (p65) and the Giles gp9 capsid though overall sequence alignment is poor. Despite containing just 57 amino acids, lambda NinD was found by Blasche et al. (2013) to participate in at least 19 different PPI with *E. coli* proteins. This domain could be a promiscuous interactor, at least. Similarly, a region of phage T4 gp56, a 171 amino acid dCTPase, shares some sequence similarity with the Giles gp25 virion protein.

Structural phage proteins appear frequently among the set of phage-host interactions. Capsid proteins, including Gp62 of *Pseudomonas* phage LUZ24, GP32 of *Pseudomonas* phage LUZ19, G8P of *Pseudomonas* phage Pf3, Gp10 of *Enterobacteria* phage T7, and Gp5 of *Enterobacteria* phage T7 have all been experimentally observed to interact with chaperones (e.g., GroEL), membrane-bound proteins (e.g., YidC), or DNA-binding proteins (e.g., *Pseudomonas* MvaT; or the DNA polymerase beta subunit, DnaN). Though these interactions do not involve orthologs of the same one or two proteins, all of these host proteins have crucial roles in protein folding and transport, DNA replication, or transcriptional regulation (in the case of MvaT, this protein is a global

repressor associating with about 110 different chromosomal regions, as per Castang et al. (2008)).

As a general (yet, due to its involvement with a species beyond *E. coli* or *P. aeruginosa*, some what exotic) example, I have Corynephage BFK20 Gp41 helicase. Solteszova et al. (2015) found interactions between this protein and host proteins DnaZX, DnaN, Dnaδ, DnaG and SSB using bacterial two hybrid. The replication proteins are conserved in a variety of other species but this particular interaction may or not be preserved; Bacillus phage SPP1 G40P helicase was also found to interact with Bacillus DnaG by Wang et al. (2008) and by Ayora et al. (1998). This interaction holds importance for our understanding of the varied mechanisms of DNA replication. In phage terms, it establishes the types of host proteins most frequently involved in type 2 phage replication (that is, those encoding DNA polymerase components; this concept is reviewed in great detail by Weigel and Seitz (2006)). Phages are frequently used as models of DNA replication as far back as the establishment of the function of DNA and the proteins involved in this process have been well-studied but vary across phages. It is therefore crucial to have a baseline of exact protein-protein interactions to work with for consistency.

An integrative approach to interactomes of phage vs. phage and phage vs. host PPI is methodologically limited in certain critical ways. In some cases, such as with the Giles interactome, the protein sequences involved are simply too different from most known

sequences for more than speculative functional inference to be made about them. Giles was the first mycobacteriophage to be subject to full interactome screening, however, and it remains possible, that similar interaction patterns may emerge should even distantly related phages receive the same treatment. Perhaps more importantly, despite more than 50 years of excellent phage research, the field is only recently devoting focus to direct PPI between virus and host beyond those interactions most essential to infection. The vast genetic diversity within phage genomes will surely offer novel possibilities for informative interactions.

4.4.2 A curated set of phage-host PPI

The set of curated bacteriophage vs. bacterial protein-protein interactions (see **Appendix Table V-A**) offers three immediate benefits. It provides a single compilation of the published, experimental observations of a specific class of common protein interactions. Though at least one review has pursued a similar goal (Häuser et al. 2012), it offers a smaller set of direct, binary interactions and does not include the results of several large-scale phage vs. host interaction screens performed in the last several years. This specific qualifier of “binary” interactions is in contrast to the more generalized observations determined using mutational studies. While such studies continue to offer compelling data for biological phenomena, they leave open the possibility of secondary effects. Finally, application of orthology-driven methods for protein and interaction comparison (as seen in Chapters 2 and 3; see the Methods of each respective section for further details) allows the database to serve as a method for

comparing interactions. Rather than existing as isolated sets of protein interactions, relationships may be expressed as interactions between orthologous groups.

General properties of the set of phage-host interactions are provided in **Table 4-A**. The set contains 254 proteins of unique sequence in total, corresponding to representatives from 29 different phages and 10 different bacterial species (**Table 4-A**). In total, results from 48 publications (**Appendix Table V-B**) are included in the set, or just over 6 interactions for each study (though some studies contribute many more interactions than others, e.g. work by Van den Bossche et al. (2014) contributes 80 PPI and a study by Blasche et al. (2013) contributes 103 PPI). Some phage proteins (e.g., lambda G protein) also contribute many more interactions than others (in this case, 12 PPI involve lambda G), potentially due to their inclusion in large-scale studies with more opportunities to observe novel PPI.

Table 4-A. Descriptive statistics of the phage-host protein interaction data set.

Property	Value
Unique proteins	254
Unique proteins, from phage	121
Unique proteins, from bacteria	133
Interactions	294
Unique phages	29
Unique bacterial species	10
Publications	48
Unique OGs, from phage*	120 (106)
Unique OGs, from bacteria*	126 (2)
Average interactions per phage	5.73 +/- 5.51

protein	
Average interactions per bacterial protein	6.00 +/- 6.84

* Number in parenthesis is proteins without OG annotation; these are counted as single-member OGs.

As with any set of combined biological data from experimental sources, the set of phage vs. host interactions is primarily defined by the most commonly studied subjects. **Fig. 4-B** provides a breakdown of the data by bacterial species and by virus serving as a source of proteins found participating in interactions. Interactions with proteins from *E. coli* phages contribute ~47% of the interactions, particularly those from phage lambda (105 interactions, or ~36% of the interactions) (**Fig. 4-B-A**). The phages T4 and T7 contribute another 13 and 11 PPI, respectively. Interestingly, the *Pseudomonas* phage YuA contributes 38 PPI, as many as *Streptococcus* phage Dp-1, though the *Pseudomonas* phage was screened with a much smaller set of host proteins than in the whole-host-genome screen performed by Mariano et al. (2016) with phage Dp-1. As expected, the host bacterial species contributing the most PPI is dominated by the hosts of the phages mentioned above: *E. coli*, *Pseudomonas aeruginosa*, and *Streptococcus pneumoniae* (**Fig. 4-B-B**). Just 18 PPI in the set involve proteins from other bacterial species.

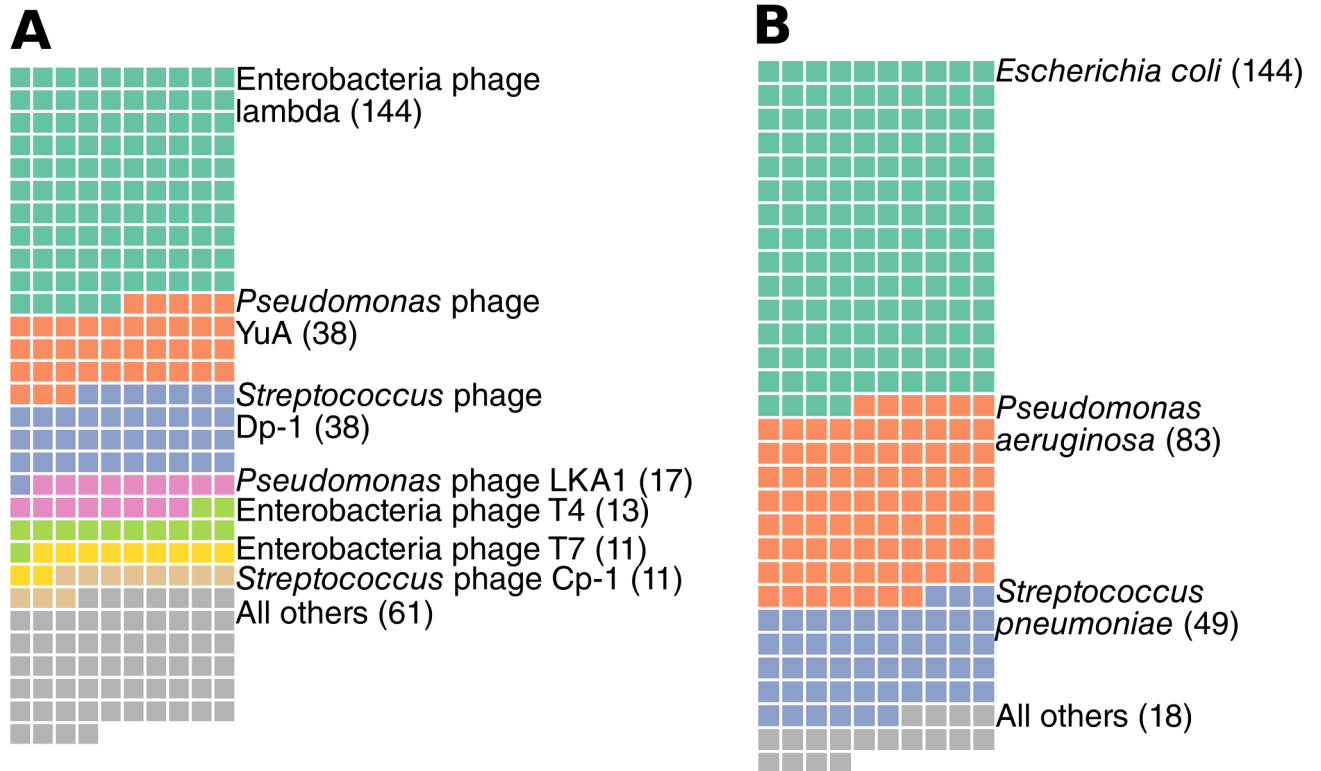


Fig. 4-B. Composition of the observed phage vs. host protein-protein interactions by phage or host. Each box represents a single protein-protein interaction. Counts in parentheses are protein-protein interactions. **A)** Contributions of interactors from specific phages. **B)** Contributions of interactors from specific bacterial species.

4.4.3 A network and meta-network analysis of phage-host PPI

Fig. 4-C provides the full set of phage-host PPI as a network. This visualization reveals differences in structure between subsets of the data. The cluster of *E. coli* interactions seen at the left corresponds to phage lambda interactions. Phage lambda proteins have been observed (largely through the high-throughput yeast two hybrid screens by Blasche et al. (2013)) interacting with numerous host proteins. The *Pseudomonas* phage-host interactions in the neighboring cluster demonstrate a different pattern: here, a set of nine host proteins interacts with many more phage proteins, primarily reflecting the experimental design employed by Van den Bossche et al. (2014). Unlike the lambda vs. host screens, the Van den Bossche et al. *Pseudomonas* phage screens involved no host proteins beyond the nine screened.

Much of this network is fragmented. In most cases, bacteriophage proteins have not been observed interacting with host proteins from multiple bacterial species, though this is likely because the screens have not been performed. A single exception to the rule is shown with a phage protein interacting with proteins from both *E. coli* and *Pseudomonas*. This interaction is that of phage T4 AsiA; Dove and Hochschild (2001) found this protein to interact with RNA polymerase Sigma-70 from both bacterial species. Otherwise, most phage proteins have been found to interact with either one or two host proteins, though in some cases the same host protein has been found to interact with multiple phage proteins.

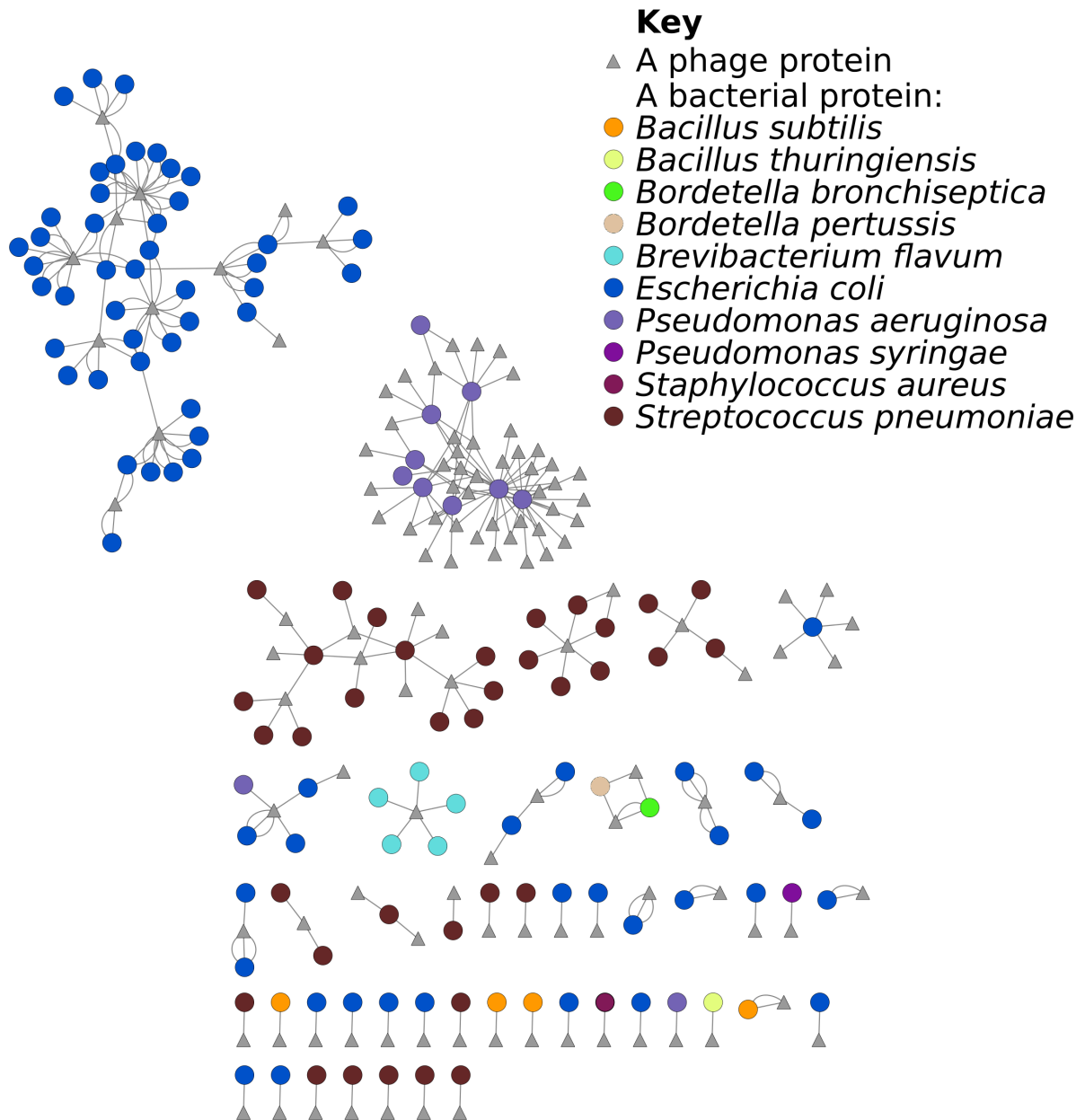


Fig. 4-C. The network of phage vs. host protein-protein interactions. Here, all nodes are unique proteins and all edges indicate an interaction observed between two proteins; multiple edges between nodes indicate observation of an interaction by multiple methods and/or studies. Nodes representing bacterial proteins are colored by species of origin.

Figure 4-D provides the same set of interactions as shown in **Fig. 4-C**, though in this case the protein interactors have been compressed into orthologous groups (OGs) wherever possible. As noted above in **Table 4-A**, this has the most effect on bacterial interactors, as just 14 of 120 phage interactors can be mapped to OGs. To maximize the impact of orthology-based comparisons, the construction of this network also filters out all interactors participating in only single interactions. This compression has the greatest impact on bacterial proteins as the viral protein interactors demonstrate much less sequence similarity and, as a general result, either do not map to OGs or do not share OG membership. Even so, this presentation of the network reveals novel consistencies among phage-host interactions. In all, this network contains 268 interactions among 194 interactors. Many interactions are compressed as a similar type of interaction was observed multiple times in the same data set. The cluster of interactions formerly dominated by *Pseudomonas* interactions is revealed to involve similar types of interactions in *Brevibacterium flavum* and *E. coli*. Similarly, many of the formerly single phage protein vs. single host protein interactions have now either been filtered out (as noted above, due to their participation in just one interaction and hence little value for interaction pattern prediction) or have been compressed into larger, more interconnected clusters, especially in the case of *E. coli* interactions.

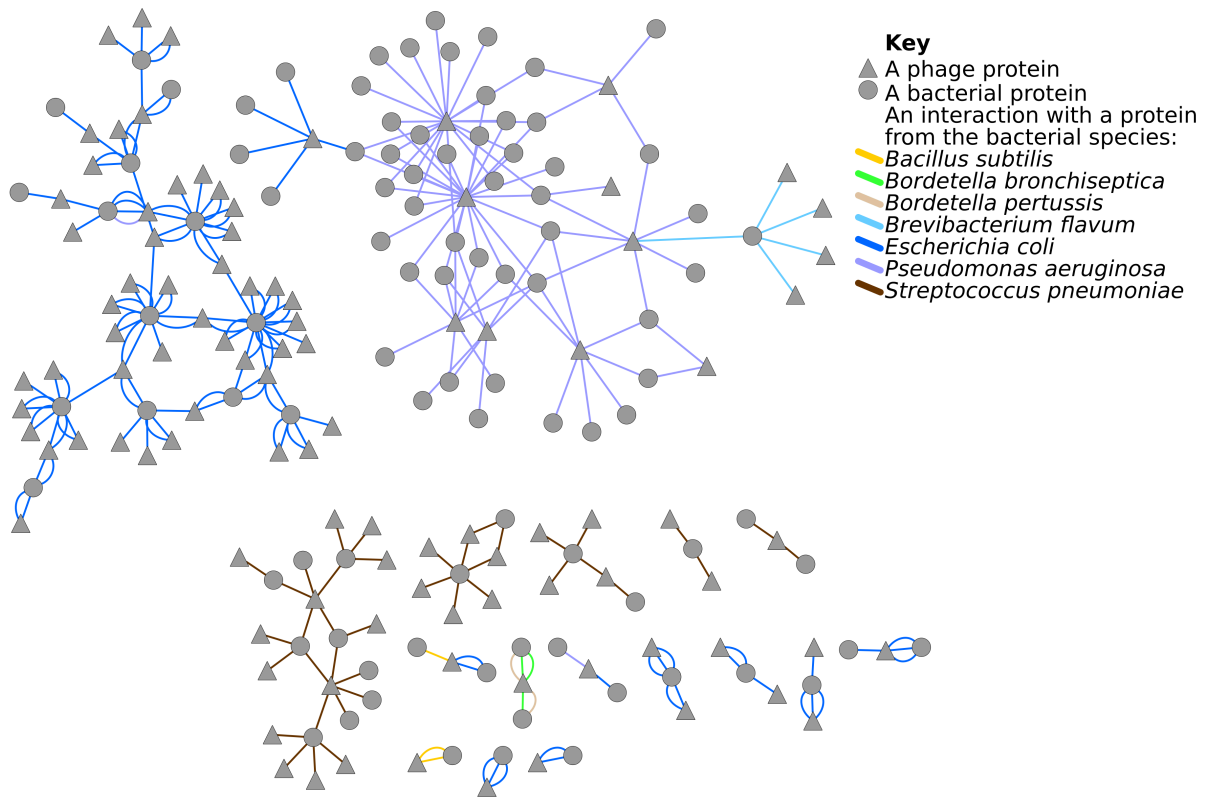


Fig. 4-D. The meta-network of multiple-incidence phage vs. host protein-protein interactions. Here, all nodes are unique OGs (here, this indicates either an eggNOG OG or treatment of a unique protein as a single-member OG, as is the case with most phage proteins) and all edges indicate an interaction observed between two OGs; multiple edges between nodes indicate observation of an interaction by multiple methods and/or studies. Edges are colored by bacterial species of origin of protein interactors. In this network, all interactions involve OGs participating in more than one interaction.

Some of the benefit of an OG-focused approach to PPI networks can be seen with individual data sets. Among the 102 PPI identified by Blasche et al. (2013) between lambda and *E. coli* proteins, for example, the 133 host proteins involved in these PPI can be compressed into 47 OGs. Some groups, such as ENOG4105FKG (an OG containing genes for phage-like Noh terminases and DNA packaging proteins) appear to be almost completely restricted to *E. coli* genomes. Other groups, such as ENOG4105CF7 (in *E. coli*, the uncharacterized transcriptional regulator YqhC) appear to be conserved across hundreds of other bacterial genomes (though in this case, primarily in the Proteobacteria). Converting the lambda proteins in these PPI to OGs does not reduce the total number of interactors – it remains 15 in both cases – but does allow for cross-phage comparisons with proteins not seen in the network, as with host proteins. The lambda protein NinD maps to ENOG411EP1E, for instance, an member of which is also present in the genome of fellow enterobacteria phage P22. Phage-host interaction screens have not been performed using phage P22, though interestingly, comparisons between lambda and P22 have revealed distinctive differences in phage genome construction (Gough and Levine 1968) yet enough similarity to allow the two viruses to hybridize with each other (albeit under carefully engineered conditions, see Botstein and Herskowitz 1974).

4.4.4 Phage-host PPI involve broadly-conserved protein complex components

Perhaps the most striking consistency among the PPI in the set of phage-host interactions is the prevalence of interactions with protein complex components. As shown previously in Chapter 2, though *E. coli* protein complexes are a rough model for bacterial protein complexes as a whole, they provide a general concept of the potential secondary impacts of a PPI involving a protein complex component. Out of all 294 phage-host interactions, 55 involve a host protein either present in an *E. coli* protein complex or with an ortholog in one. As these complexes do not fully capture the full extent of orthology, I may also include proteins of more distant sequence similarity in the set of protein complex components (e.g., *Pseudomonas* DNA polymerase components are not fully orthologous to those of *E. coli* but likely provide identical functions). With this adjustment, the number of phage-host PPI involving a protein complex component increases to 93.

Fig. 4-E provides a network of interactions predicted to occur between phages and protein complexes. Each interaction in this network is derived from at least one PPI between a phage protein and a host protein component in at least one pair of virus and host. Here, the protein complexes are defined by those in the EcoCyc set of literature-curated *E. coli* complexes (see **Chapter 3** Methods). As 24 of the 67 interactions shown in this network involve bacterial species other than *E. coli*, membership in the same OG *or a homologous protein complex* is considered sufficient for protein complex membership. For example, *P. aeruginosa* RpoD is a member of ENOG4105DG1, as is *E. coli* RpoD. All interactions involving ENOG4105DG1 are therefore considered to be

interactions with RNA polymerase (though, in this network, RNA polymerase is specifically the holoenzyme containing RpoD, while the holoenzyme containing the stationary phase sigma38 factor RpoS is defined separately).

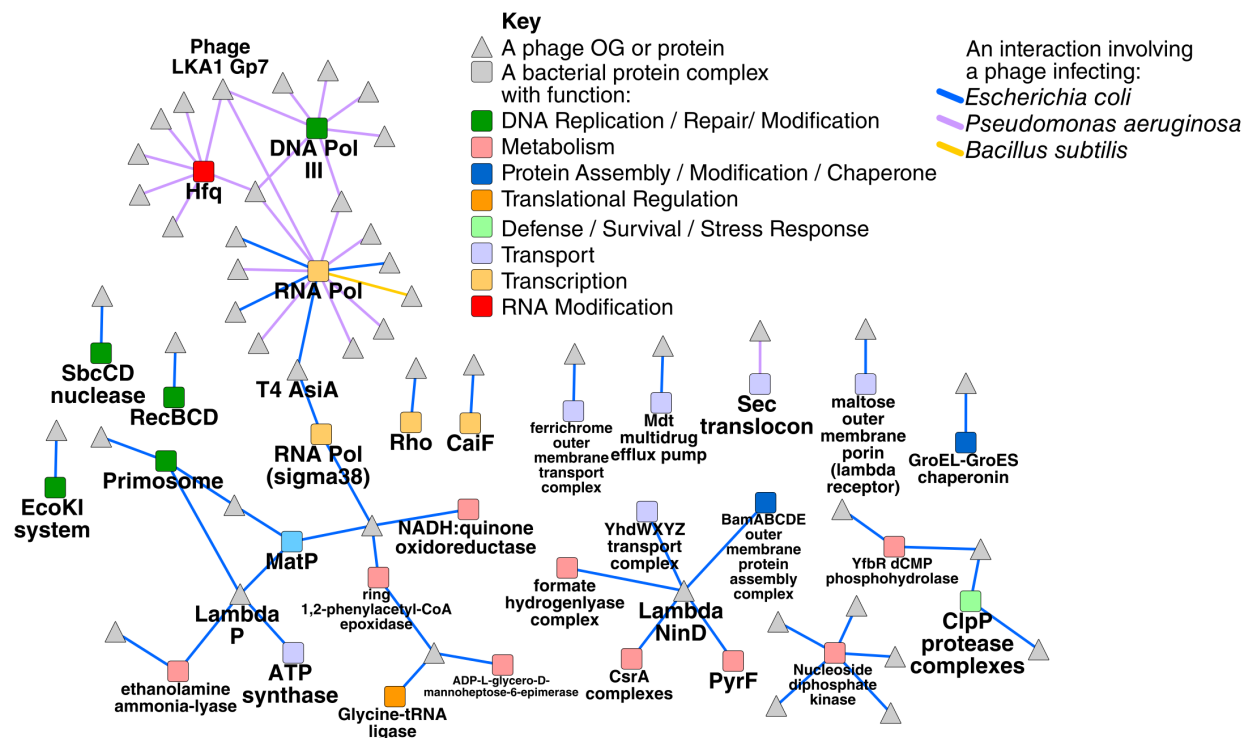


Fig. 4-E. The meta-network of phage protein vs. host complex interactions. Here, all bacteriophage nodes are unique OGs (here, this indicates either an eggNOG OG or treatment of a unique protein as a single-member OG, as is the case with most phage proteins), all host nodes are unique protein complexes as defined by EcoCyc, and all edges indicate *any* interaction observed between two OGs in which one interactor is a protein complex component. Edges denote origin of bacteriophage interactor and are predicted interactions in those involving phages with hosts other than *E. coli*. Bacteriophage node labels partially omitted for clarity; see **Appendix Table V-A** for full set of interactions.

At least 10 different phage proteins have been found to interact components of polymerases. In *Pseudomonas* phages, the interactors include 17 different proteins from phage YuA, two from phage ϕ KZ, and one from phage PEV interacting with DNA polymerase components specifically. *E. coli* phage T4 AsiA has also been observed in interactions with polymerase subunits, though its role has been more closely studied: AsiA is a transcriptional inhibitor, actively weakens the interaction between host sigma70 and RNA polymerase (Lambert et al. 2004), and is necessary for T4 transcription (Ouhammouch et al. 1994). Lambda N protein has also been found to interact with RpoD along with a variety of other proteins, including RpoS (Blasche et al. 2013). These interactions may be biased with respect to other proteins as DnaX was specifically identified by Van den Bossche et al. (2014) as a likely interactor with *Pseudomonas* phage proteins, though the interaction likely holds biological relevance as DNA replication is a likely target during the phage life cycle, other host DNA replication proteins are implicated in PPI, and bacteriophages lacking their own polymerases are known to use the host proteins for their own replication (reviewed for phage lambda by Skalka 1977).

Perhaps the most noticeable result of a network analysis is the potential for cross-functional interactions. Lambda NinD, in particular, appears to interact with different sets of host proteins, including the formate hydrogenlyase complex and the outer membrane protein assembly complex BamABCDE, and an uncharacterized amino acid transporter, YhdWXYZ. Lambda replication protein P appears to interact with yet a different set of

complexes, including part of ATP synthase, ethanolamine ammonia-lyase, and the primosome.

4.4.5 Additional discussion and future work

The concept of the interactome is compelling as it implies activity. Whereas a genome or a proteome is essentially a list or census of a cell's prospective parts, an interactome maps the potential for interplay between those parts. In this sense, an interactome attempts to comprehensively model biomolecular activity (or, at least, assumes protein-protein interactions play a role) within a cell. A bacterial cell's existence is rarely static, however. It must respond to stressors and threats in its environment, or in the case of bacteriophage infection, threats within the confines of its own membranes. In this stage of my research, I have attempted to define the state of human knowledge of this type of relationship within the framework of interactomics.

Through careful curation and application of orthology comparisons, I assembled a set of more than 290 phage vs. host protein-protein interactions (PPI) and determined commonalities among these PPI. Nearly a third of these interactions involve protein complex components, often of complexes crucial to bacterial life and replication (e.g., RNA and DNA polymerases or ribosomal proteins). As a resource, this data set may be most useful when interpreted along with binary phage vs. phage protein interactions or even those from full bacterial interactomes.

The data set provided here should ideally serve as a starting point for further study into phage-host protein interactions. As a resource, it not only provides a set of binary interactions specific to the field of phage vs. host interactions, but serves as a guide for

the interactions found in future interaction screens. These interactions are complementary to and improved by recently released bacteriophage protein orthology databases (Kristensen et al. 2013; Grazziotin et al. 2016). Researchers may use interactors and orthology-mapped interactions to rapidly determine whether they have duplicated previous findings in new interaction screens. Furthermore, such a resource allows for consistency in interpreting future interactomics studies, even those involving no viral proteins. Researchers studying bacterial PPIs and interactomes may consult this resource to gauge the likelihood that bacteriophage proteins may compete with host proteins in interactions.

Future studies into phage vs. host interactions may offer some of the few chances to discover novel antibacterial treatments. The most effective strategies for controlling bacterial infections may already exist within a phage's genome. In order for a treatment to be truly effective against the assortment of pathogens present in an infection, however, such a treatment must be designed to target multiple species and should involve a cocktail of phage subtypes (Levin and Bull 2004, Chan et al. 2013, Mattila et al. 2015). The full set of potential protein interactions should be kept in mind. An orthology-based, cross-species approach to phage-host PPIs should therefore become standard in all investigations of phage therapy.

Chapter 5 – Conclusions

5.1 Protein complexes are irregularly conserved across divergent bacterial species

As part of the work presented here, I first coupled the results of mass spectrometry-characterized protein complexes (Hu et al. 2009, Kühner et al. 2009) with databases of gene orthology (Powell et al. 2012) and essentiality (Luo et al. 2014) to characterize interaction conservation within protein complexes. Furthermore, I used the perspective of genome reduction to evaluate patterns across levels of protein conservation. Comparing sets of protein complexes from divergent bacterial species (in this case, *E. coli* and *M. pneumoniae*) alleviates some of the bias inherent in using a single species as a universal model. Rather, observing which protein complexes and their components are present in two otherwise distinct species allows us to draw conclusions about how crucial these components are to bacterial life.

5.2 A protein-protein meta-interactome provides context for conserved interactions

Next, I combined experimentally-derived, previously published protein-protein interactions from 349 bacterial species to form a consensus meta-interactome. This approach uses orthologous groups (OG) of proteins to combine all known interactions into a single network. Notably, I observed that such a network shares characteristics of single species interactomes. Furthermore, the augmentation of single species interaction networks with a bacterial meta-interactome improves its efficacy in predicting

functions of the underlying proteins, given its dramatically increased information content. Finally, I used such a bacterial meta-interactome to predict interactome sizes of species for which incomplete interaction data is available.

5.3 A curated set of phage-host protein interactions provides a starting point for phage-host interactome screens

Finally, I assembled and curated a set of experimentally-verified interactions between proteins from 10 bacterial species and 29 bacteriophages, sourced from 48 different studies. Unlike previous studies of phage vs. host interactions, I have specifically focused on direct protein-protein interactions rather than indirect mutational comparisons or predicted impacts on phage infectivity. I have used this new resource to further analyze the most frequently observed types of protein interactions between bacteria and their viruses. These resources will ideally serve as a basis for further study into bacteria vs. bacteriophage interactions, bacterial protein function, and potential targets for antibacterial or bacteriostatic phage therapy.

5.3 Future work

The work done for this project will be improved by making it more biologically relevant *and* by making it more researcher-relevant. The three foci of this work rely heavily upon orthology predictions and as a result are unavoidably biased by how orthologous groups are assembled. Currently, group definitions are based on full protein sequences, potentially creating false-positive group membership when sequences are similar but functions diverge. A domain-based approach in which groups are composed of proteins

with shared domains and sequence motifs may better reveal interaction patterns. Such an approach may avoid the problem of assigning proteins with very common sequences (e.g., DNA binding domains) to the same groups but may still be subject to bias if some domains simply receive preference in annotation (Schnoes et al. 2013). It also remains difficult (but feasible, as shown by Oates et al. 2013) to ascertain when a domain will retain its structure, especially if the physiochemical context of its expression is not conducive to maintaining the domain's stability (Yegambaram et al. 2013).

In the longer term, more rigorous methods may noticeably improve the work presented here. A machine learning approach to assigning orthology or gene cluster membership may be the best option to avoid the preferential annotation bias discussed above; such methods have been attempted, though generally with genes from no more than four bacterial species at a time (Tetko et al. 2007, Plaimas et al. 2010, Škunca et al. 2013). A perfect adherence to biological relevance will be pointless, however, if its results are inaccessible. This and similar projects would benefit noticeably from installation and maintenance on a publicly-available web server. At this time, all code and data tables produced as part of this project will be made available online, though I hope future researchers may be able to improve their accessibility to others in the field of microbiology, interactomics, and beyond.

Approaching the large data sets now common among microbiology studies from a systems biology perspective may seem obvious. When faced with more genomes,

protein sequences, and protein interactions than one human could possibly interpret in a lifetime, the natural assumption is that combining data from disparate sources will naturally yield otherwise unclear correlations. As exemplified from this work, such ventures go beyond providing immediate insights: they provide conceptual paradigms and usable data for future analyses. Our understanding of the protein interactions crucial to bacterial life will undoubtedly be essential to understanding the ongoing relationship between humans and the many microbial inhabitants of our world.

REFERENCES

- Abt, M. C., McKenney, P. T. & Pamer, E. G. Clostridium difficile colitis: pathogenesis and host defence. Nat. Rev. Microbiol. 14, 609–620 (2016).
- Abu-Farha, M., Elisma, F. & Figeys, D. Identification of protein-protein interactions by mass spectrometry coupled techniques. Advances in Biochemical Engineering/Biotechnology 110, 67–80 (2008).
- Albert, R. & Barabási, A.-L. Statistical mechanics of complex networks. Rev. Mod. Phys. 74, 47–97 (2002).
- Alfarano, C. et al. The Biomolecular Interaction Network Database and related tools 2005 update. Nucleic Acids Res. 33, D418-24 (2005).
- Altschul, S. F. et al. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucleic Acids Res. 25, 3389–3402 (1997).
- Andersen, J. et al. Multidrug Efflux Pumps from Enterobacteriaceae, Vibrio cholerae and Staphylococcus aureus Bacterial Food Pathogens. Int. J. Environ. Res. Public Health 12, 1487–1547 (2015).
- Arifuzzaman, M. et al. Large-scale identification of protein-protein interaction of Escherichia coli K-12. Genome Res. 16, 686–691 (2006).
- Ayora, S., Langer, U. & Alonso, J. C. Bacillus subtilis DnaG primase stabilises the bacteriophage SPP1 G40P helicase-ssDNA complex. FEBS Lett. 439, 59–62 (1998).
- Baba, T. et al. Construction of Escherichia coli K-12 in-frame, single-gene knockout mutants: the Keio collection. Mol. Syst. Biol. 2, 2006.0008 (2006).
- Barabási, A.-L. & Albert, R. Emergence of scaling in random networks. Science (80-.). 286, 11 (1999).
- Battchikova, N., Eisenhut, M. & Aro, E.-M. Cyanobacterial NDH-1 complexes: Novel insights and remaining puzzles. Biochim. Biophys. Acta - Bioenerg. 1807, 935–944 (2011).
- Botstein, D. & Herskowitz, I. Properties of hybrids between Salmonella phage P22 and coliphage λ. Nature 251, 584–589 (1974).
- Braun, P. et al. An experimentally derived confidence score for binary protein-protein interactions. Nat. Methods 6, 91–97 (2009).

- Brown, K. R. & Jurisica, I. Online Predicted Human Interaction Database. *Bioinformatics* 21, 2076–2082 (2005).
- Calderone, A., Licata, L. & Cesareni, G. VirusMentha: a new resource for virus-host protein interactions. *Nucleic Acids Res.* 43, D588–D592 (2015).
- Callebaut, I., Mornon, J.-P., Gilgès, D. & Vigon, I. HYR, an extracellular module involved in cellular adhesion and related to the immunoglobulin-like fold. *Protein Sci.* 9, 1382–1390 (2000).
- Caraux, G. & Pinloche, S. PermutMatrix: a graphical environment to arrange gene expression profiles in optimal linear order. *Bioinformatics* 21, 1280–1281 (2005).
- Castang, S., McManus, H. R., Turner, K. H. & Dove, S. L. H-NS family members function coordinately in an opportunistic pathogen. *Proc. Natl. Acad. Sci.* 105, 18947–18952 (2008).
- Caufield, J. H., Sakhawalkar, N. & Uetz, P. A comparison and optimization of yeast two-hybrid systems. *Methods* 58, 317–324 (2012).
- Caufield, J. H., Abreu, M., Wimble, C. & Uetz, P. Protein Complexes in Bacteria. *PLOS Comput. Biol.* 11, e1004107 (2015).
- Centers for Disease Control and Prevention. Antibiotic resistance threats in the United States. Centers for Disease Control and Prevention (2013).
- Chan, B. K., Abedon, S. T. & Loc-Carrillo, C. Phage cocktails and the future of phage therapy. *Future Microbiol.* 8, 769–783 (2013).
- Chatr-Aryamontri, A. et al. The BioGRID interaction database: 2015 update. *Nucleic Acids Res.* 43, D470–D478 (2015).
- Chatterjee, P. K. & Sternberg, N. L. A general genetic approach in *Escherichia coli* for determining the mechanism(s) of action of tumoricidal agents: application to DMP 840, a tumoricidal agent. *Proc. Natl. Acad. Sci. U. S. A.* 92, 8950–4 (1995).
- Chen, C. et al. Subunit–subunit interactions in the human 26S proteasome. *Proteomics* 8, 508–520 (2008).

- Chen, Y.-C., Rajagopala, S. V., Stellberger, T. & Uetz, P. Exhaustive benchmarking of the yeast two-hybrid system. *Nat. Methods* 7, 667–668 (2010).
- Cherkasov, A. et al. Mapping the Protein Interaction Network in Methicillin-Resistant *Staphylococcus aureus*. *J. Proteome Res.* 10, 1139–1150 (2011).
- Cherkasov, A. et al. Mapping the Protein Interaction Network in Methicillin-Resistant *Staphylococcus aureus*. *J. Proteome Res.* 10, 1139–1150 (2011).
- Christen, B. et al. The essential genome of a bacterium. *Mol. Syst. Biol.* 7, 528–528 (2014).
- Cohen, O., Ashkenazy, H., Levy Karin, E., Burstein, D. & Pupko, T. CoPAP: Coevolution of presence-absence patterns. *Nucleic Acids Res.* 41, W232-7 (2013).
- Davy, A. et al. A protein-protein interaction map of the *Caenorhabditis elegans* 26S proteasome. *EMBO Rep.* 2, 821–828 (2001).
- de Matos Simoes, R., Dehmer, M. & Emmert-Streib, F. Interfacing cellular networks of *S. cerevisiae* and *E. coli*: connecting dynamic and genetic information. *BMC Genomics* 14, 324 (2013).
- Dedrick, R. M. et al. Functional requirements for bacteriophage growth: gene essentiality and expression in mycobacteriophage Giles. *Mol. Microbiol.* 88, 577–589 (2013).
- Dixon, S. J., Costanzo, M., Baryshnikova, A., Andrews, B. & Boone, C. Systematic mapping of genetic interaction networks. *Annu. Rev. Genet.* 43, 601–625 (2009).
- Dove, S. L. & Hochschild, A. Bacterial Two-Hybrid Analysis of Interactions between Region 4 of the 70 Subunit of RNA Polymerase and the Transcriptional Regulators Rsd from *Escherichia coli* and AlgQ from *Pseudomonas aeruginosa*. *J. Bacteriol.* 183, 6413–6421 (2001).
- Eraso, J. M. et al. The highly conserved MraZ protein is a transcriptional regulator in *Escherichia coli*. *J. Bacteriol.* 196, 2053–2066 (2014).
- Federhen, S. The NCBI Taxonomy database. *Nucleic Acids Res.* 40, D136-43 (2012).
- Friedel, C. C. & Zimmer, R. Inferring topology from clustering coefficients in protein-protein interaction networks. *BMC Bioinformatics* 7, 519 (2006).

- Gallos, L. K., Makse, H. A. & Sigman, M. A small world of weak ties provides optimal global integration of self-similar modules in functional brain networks. *Proc. Natl. Acad. Sci.* 109, 2825–2830 (2012).
- Gandhi, T. K. B. et al. Analysis of the human protein interactome and comparison with yeast, worm and fly interaction datasets. *Nat. Genet.* 38, 285–293 (2006).
- Glass, J. I. et al. Essential genes of a minimal bacterium. *Proc. Natl. Acad. Sci.* 103, 425–430 (2006).
- Gough, M. & Levine, M. The circularity of the phage P22 linkage map. *Genetics* 58, 161–9 (1968).
- Grazziotin, A. L., Koonin, E. V. & Kristensen, D. M. Prokaryotic Virus Orthologous Groups (pVOGs): a resource for comparative genomics and protein family annotation. *Nucleic Acids Res.* gkw975 (2016). doi:10.1093/nar/gkw975
- Gu, H., Zhu, P., Jiao, Y., Meng, Y. & Chen, M. PRIN: a predicted rice interactome network. *BMC Bioinformatics* 12, 161 (2011).
- Guerrero, C., Tagwerker, C., Kaiser, P. & Huang, L. An integrated mass spectrometry-based proteomic approach: quantitative analysis of tandem affinity-purified in vivo cross-linked protein complexes (QTAX) to decipher the 26 S proteasome-interacting network. *Mol. Cell. Proteomics* 5, 366–78 (2006).
- Guimera, R. & Sales-Pardo, M. Missing and spurious interactions and the reconstruction of complex networks. *Proc. Natl. Acad. Sci.* 106, 22073–22078 (2009).
- Guirimand, T., Delmotte, S. & Navratil, V. VirHostNet 2.0: surfing on the web of virus/host molecular interactions data. *Nucleic Acids Res.* 43, D583–D587 (2014).
- Han, K. et al. Extraordinary expansion of a *Sorangium cellulosum* genome from an alkaline milieu. *Sci. Rep.* 3, (2013).
- Hart, G. T., Lee, I. & Marcotte, E. R. A high-accuracy consensus map of yeast protein complexes reveals modular nature of gene essentiality. *BMC Bioinformatics* 8, 236 (2007).
- Häuser, R. et al. Bacteriophage Protein-Protein Interactions. *Adv. Virus Res.* 83, 219–298 (2012).
- Häuser, R. et al. A second-generation protein-protein interaction network of *Helicobacter pylori*. *Mol. Cell. Proteomics* 13, 1318–29 (2014).

- Haynes, C. et al. Intrinsic Disorder Is a Common Feature of Hub Proteins from Four Eukaryotic Interactomes. *PLoS Comput. Biol.* 2, e100 (2006).
- He, Z. & Mi, H. Functional characterization of the subunits N, H, J, and O of the NAD(P)H dehydrogenase complexes in *Synechocystis* sp. strain PCC 6803. *Plant Physiol.* pp.00458.2016 (2016). doi:10.1104/pp.16.00458
- Helke, K. L. et al. Effects of antimicrobial use in agricultural animals on drug-resistant foodborne salmonellosis in humans: A systematic literature review. *Crit. Rev. Food Sci. Nutr.* 57, 472–488 (2016).
- Helms, M., Simonsen, J. & Mølbak, K. Quinolone Resistance Is Associated with Increased Risk of Invasive Illness or Death during Infection with *Salmonella* Serotype Typhimurium. *J. Infect. Dis.* 190, 1652–1654 (2004).
- Hermjakob, H. et al. The HUPO PSI's Molecular Interaction format—a community standard for the representation of protein interaction data. *Nat. Biotechnol.* 22, 177–183 (2004).
- Holt, K. E. et al. Genomic analysis of diversity, population structure, virulence, and antimicrobial resistance in *Klebsiella pneumoniae*, an urgent threat to public health. *Proc. Natl. Acad. Sci.* 112, E3574–E3581 (2015).
- Hu, P. et al. Global functional atlas of *Escherichia coli* encompassing previously uncharacterized proteins. *PLoS Biol.* 7, 0929–0947 (2009).
- Huerta-Cepas, J. et al. eggNOG 4.5: a hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences. *Nucleic Acids Res.* (2015). doi:10.1093/nar/gkv1248
- Ito, T. et al. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc. Natl. Acad. Sci. U. S. A.* 98, 4569–4574 (2001).
- Kamran, M., Sinha, S., Dubey, P., Lynn, A. M. & Dhar, S. K. Identification of putative Z-ring-associated proteins, involved in cell division in human pathogenic bacteria *Helicobacter pylori*. *FEBS Lett.* 590, 2158–2171 (2016).

- Kanehisa, M. & Goto, S. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* 28, 27–30 (2000).
- Katahira, K., Ogura, Y., Gotoh, Y. & Hayashi, T. Draft Genome Sequences of Five Rapidly Growing *Mycobacterium* Species, *M. thermoresistibile*, *M. fortuitum* subsp. *acetamidolyticum*, *M. canariasense*, *M. brisbanense*, and *M. novocastrense*: TABLE 1. *Genome Announc.* 4, e00322–16 (2016).
- Kelkar, Y. D. & Ochman, H. Genome reduction promotes increase in protein functional complexity in bacteria. *Genetics* 193, 303–307 (2013).
- Kembel, S. W. et al. Picante: R tools for integrating phylogenies and ecology. *Bioinformatics* 26, 1463–1464 (2010).
- Kerrien, S. et al. IntAct--open source resource for molecular interaction data. *Nucleic Acids Res.* 35, D561–D565 (2007).
- Kerrien, S. et al. The IntAct molecular interaction database in 2012. *Nucleic Acids Res.* 40, D841–6 (2012).
- Keseler, I. M. et al. EcoCyc: Fusing model organism databases with systems biology. *Nucleic Acids Res.* 41, D605–12 (2013).
- Kim, M.-S. et al. A draft map of the human proteome. *Nature* 509, 575–581 (2014).
- Kobayashi, K. et al. Essential *Bacillus subtilis* genes. *Proc. Natl. Acad. Sci.* 100, 4678–4683 (2003).
- Koonin, E. V, Mushegian, a R. & Bork, P. Non-orthologous gene displacement. *Trends Genet.* 12, 334–336 (1996).
- Korste, A., Wulforth, H., Ikegami, T., Nowaczyk, M. M. & Stoll, R. Solution structure of the NDH-1 complex subunit CupS from *Thermosynechococcus elongatus*. *Biochim. Biophys. Acta - Bioenerg.* 1847, 1212–1219 (2015).
- Kotlyar, M. et al. In silico prediction of physical protein interactions and characterization of interactome orphans. *Nat. Methods* 12, 79–84 (2014).

- Kristensen, D. M. et al. A low-polynomial algorithm for assembling clusters of orthologous groups from intergenomic symmetric best matches. *Bioinformatics* 26, 1481–7 (2010).
- Kristensen, D. M. et al. Orthologous gene clusters and taxon signature genes for viruses of prokaryotes. *J. Bacteriol.* 195, 941–950 (2013).
- Kühner, S. et al. Proteome organization in a genome-reduced bacterium. *Science* 326, 1235–1240 (2009).
- Land, M. et al. Insights from 20 years of bacterial genome sequencing. *Funct. Integr. Genomics* 15, 141–61 (2015).
- Lasker, K. et al. Molecular architecture of the 26S proteasome holocomplex determined by an integrative approach. *Proc. Natl. Acad. Sci. U. S. A.* 109, 1380–7 (2012).
- Lee, S.-A. et al. Ortholog-based protein-protein interaction prediction and its application to inter-species interactions. *BMC Bioinformatics* 9, S11 (2008).
- Letunic, I. & Bork, P. Interactive Tree of Life v2: Online annotation and display of phylogenetic trees made easy. *Nucleic Acids Res.* 39, W475–8 (2011).
- Levin, B. R. & Bull, J. J. Opinion : Population and evolutionary dynamics of phage therapy. *Nat. Rev. Microbiol.* 2, 166–173 (2004).
- Li, S. et al. A map of the interactome network of the metazoan *C. elegans*. *Science* 303, 540–3 (2004).
- Liang, Z., Xu, M., Teng, M. & Niu, L. Comparison of protein interaction networks reveals species conservation and divergence. *BMC Bioinformatics* 7, 457 (2006).
- Liberati, N. T. et al. An ordered, nonredundant library of *Pseudomonas aeruginosa* strain PA14 transposon insertion mutants. *Proc. Natl. Acad. Sci.* 103, 2833–2838 (2006).
- Licata, L. et al. MINT, the molecular interaction database: 2012 update. *Nucleic Acids Res.* 40, D857–D861 (2012).
- Liechti, G. & Goldberg, J. B. Outer membrane biogenesis in *Escherichia coli*, *Neisseria meningitidis*, and *Helicobacter pylori*: paradigm deviations in *H. pylori*. *Frontiers in Cellular and Infection Microbiology* 2, 29 (2012).

- Luo, H., Lin, Y., Gao, F., Zhang, C. T. & Zhang, R. DEG 10, an update of the database of essential genes that includes both protein-coding genes and noncoding genomic elements. *Nucleic Acids Res.* 42, D574-80 (2014).
- Margolin, W. Sculpting the Bacterial Cell. *Curr. Biol.* 19, R812–R822 (2009).
- Matthews, L. R. et al. Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or ‘interologs’. *Genome Res.* 11, 2120–6 (2001).
- Mattila, S., Ruotsalainen, P. & Jalasvuori, M. On-Demand Isolation of Bacteriophages Against Drug-Resistant Bacteria for Personalized Phage Therapy. *Front. Microbiol.* 6, (2015).
- McCutcheon, J. P., McDonald, B. R. & Moran, N. A. Origin of an Alternative Genetic Code in the Extremely Small and GC–Rich Genome of a Bacterial Symbiont. *PLoS Genet.* 5, e1000565 (2009).
- Mehla, J. et al. The protein interactome of mycobacteriophage Giles predicts functions for unknown proteins. *J. Bacteriol.* JB.00164-15 (2015). doi:10.1128/JB.00164-15
- Morris, P., Marinelli, L. J., Jacobs-Sera, D., Hendrix, R. W. & Hatfull, G. F. Genomic characterization of mycobacteriophage giles: Evidence for phage acquisition of host DNA by illegitimate recombination. *J. Bacteriol.* 190, 2172–2182 (2008).
- Morrison, R. B. The effect of temperature and chloramphenicol on the development of flagella and motility in a strain of *Escherichia coli*. *J. Pathol. Bacteriol.* 82, 189–92 (1961).
- Mushegian, A. R. & Koonin, E. V. A minimal gene set for cellular life derived by comparison of complete bacterial genomes. *Proc. Natl. Acad. Sci. U. S. A.* 93, 10268–73 (1996).
- Nagakubo, S., Nishino, K., Hirata, T. & Yamaguchi, A. The putative response regulator BaeR stimulates multidrug resistance of *Escherichia coli* via a novel multidrug exporter system, MdtABC. *J. Bacteriol.* 184, 4161–7 (2002).
- Oates, M. E. et al. D2P2: database of disordered protein predictions. *Nucleic Acids Res.* 41, D508–D516 (2013).
- Oksanen, J. *vegan: Community Ecology Package.* (2013).

- Orchard, S. et al. The MIntAct project—IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Res.* 42, D358–D363 (2014).
- Osterman, A. et al. The Hepatitis E virus intraviral interactome. *Sci. Rep.* 5, 13872 (2015).
- Ottemann, K. M. & Lowenthal, A. C. *Helicobacter pylori* Uses Motility for Initial Colonization and To Attain Robust Infection. *Infect. Immun.* 70, 1984–1990 (2002).
- Ouhammouch, M., Orsini, G. & Brody, E. N. The *asiA* gene product of bacteriophage T4 is required for middle mode RNA synthesis. *J. Bacteriol.* 176, 3956–65 (1994).
- Parrish, J. R. et al. A proteome-wide protein interaction map for *Campylobacter jejuni*. *Genome Biol.* 8, R130 (2007).
- Plaimas, K., Eils, R. & König, R. Identifying essential genes in bacterial metabolic networks with machine learning methods. *BMC Syst. Biol.* 4, 56 (2010).
- Pope, W. H. et al. Genomics and Proteomics of Mycobacteriophage Patience, an Accidental Tourist in the *Mycobacterium* Neighborhood. *MBio* 5, e02145-14 (2014).
- Powell, S. et al. eggNOG v4.0: nested orthology inference across 3686 organisms. *Nucleic Acids Res.* 42, D231-9 (2014).
- Powell, S. et al. eggNOG v3.0: Orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Res.* 40, D284-9 (2012).
- Rain, J.-C. et al. The protein–protein interaction map of *Helicobacter pylori*. *Nature* 409, 211–215 (2001).
- Rajagopala, S. V., Casjens, S. & Uetz, P. The protein interaction map of bacteriophage lambda. *BMC Microbiol.* 11, 213 (2011).
- Rajagopala, S. V. et al. The binary protein-protein interaction landscape of *Escherichia coli*. *Nat. Biotechnol.* 32, 285–290 (2014).
- Reed, W. J. A Brief Introduction to Scale-Free Networks. *Nat. Resour. Model.* 19, 3–14 (2008).
- Rose, P. W. et al. The RCSB Protein Data Bank: New resources for research and education. *Nucleic Acids Res.* 41, D475-82 (2013).

- Ryan, C. J., Krogan, N. J., Cunningham, P. & Cagney, G. All or nothing: Protein complexes flip essentiality between distantly related eukaryotes. *Genome Biol. Evol.* 5, 1049–1059 (2013).
- Ryan, C. J. et al. Hierarchical Modularity and the Evolution of Genetic Interactomes across Species. *Mol. Cell* 46, 691–704 (2012).
- Salama, N. R., Shepherd, B. & Falkow, S. Global Transposon Mutagenesis and Essential Gene Analysis of *Helicobacter pylori*. *J. Bacteriol.* 186, 7926–7935 (2004).
- Sambourg, L. & Thierry-Mieg, N. New insights into protein-protein interaction data lead to increased estimates of the *S. cerevisiae* interactome size. *BMC Bioinformatics* 11, 605 (2010).
- Sato, S. et al. A large-scale protein-protein interaction analysis in *synechocystis* sp. PCC6803. *DNA Res.* 14, 207–216 (2007).
- Schad, E., Tompa, P. & Hegyi, H. The relationship between proteome size, structural disorder and organism complexity. *Genome Biol.* 12, R120 (2011).
- Schauer, K. & Stingl, K. 'Guilty by association' - Protein-protein interactions (PPIs) in bacterial pathogens. *Genome Dynamics* 6, 48–61 (2009).
- Schoes, A. M., Ream, D. C., Thorman, A. W., Babbitt, P. C. & Friedberg, I. Biases in the Experimental Annotations of Protein Function and Their Effect on Our Understanding of Protein Function Space. *PLoS Comput. Biol.* 9, e1003063 (2013).
- Shannon, P. et al. Cytoscape: A software Environment for integrated models of biomolecular interaction networks. *Genome Res.* 13, 2498–2504 (2003).
- Shanson, D. C. Antibiotic-resistant *Staphylococcus aureus*. *J. Hosp. Infect.* 2, 11–36 (1981).
- Sharan, R. et al. Conserved patterns of protein interaction in multiple species. *Proc. Natl. Acad. Sci.* 102, 1974–1979 (2005).
- Sharan, R., Ideker, T., Kelley, B., Shamir, R. & Karp, R. M. Identification of Protein Complexes by Comparative Analysis of Yeast and Bacterial Protein Interaction Data. *J. Comput. Biol.* 12, 835–846 (2005).

- Shimoda, Y. et al. A large scale analysis of protein-protein interactions in the nitrogen-fixing bacterium *Mesorhizobium loti*. *DNA Res.* 15, 3–11 (2008).
- Shoji, S., Dambacher, C. M., Shajani, Z., Williamson, J. R. & Schultz, P. G. Systematic Chromosomal Deletion of Bacterial Ribosomal Protein Genes. *J. Mol. Biol.* 413, 751–761 (2011).
- Sievers, F. et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.* 7, 539–539 (2011).
- Sissi, C. & Palumbo, M. Effects of magnesium and related divalent metal ions in topoisomerase structure and function. *Nucleic Acids Res.* 37, 702–711 (2009).
- Skalka, A. M. in *Current Topics in Microbiology and Immunology* 201–237 (1977). doi:10.1007/978-3-642-66800-5_7
- Škunca, N. et al. Phyletic Profiling with Cliques of Orthologs Is Enhanced by Signatures of Paralogy Relationships. *PLoS Comput. Biol.* 9, e1002852 (2013).
- Song, J. & Singh, M. How and when should interactome-derived clusters be used to predict functional modules and protein function? *Bioinformatics* 25, 3143–3150 (2009).
- Song, Y. C. et al. FlaC, a protein of *Campylobacter jejuni* TGH9011 (ATCC43431) secreted through the flagellar apparatus, binds epithelial cells and influences cell invasion. *Mol. Microbiol.* 53, 541–53 (2004).
- Stellberger, T. et al. Improving the yeast two-hybrid system with permutated fusions proteins: the Varicella Zoster Virus interactome. *Proteome Sci.* 8, 8 (2010).
- Stumpf, M. P. H. et al. Estimating the size of the human interactome. *Proc. Natl. Acad. Sci.* 105, 6959–6964 (2008).
- Szklarczyk, D. et al. STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res.* 43, D447–D452 (2014).
- Tatusov, R. L., Koonin, E. V & Lipman, D. J. A genomic perspective on protein families. *Science* 278, 631–7 (1997).

- Tatusov, R. L. et al. The COG database: an updated version includes eukaryotes. *BMC Bioinformatics* 4, 41 (2003).
- Tavernier, J. et al. MAPPIT: a cytokine receptor-based two-hybrid method in mammalian cells. *Clin. Exp. Allergy* 32, 1397–1404 (2002).
- Tetko, I. V., Rodchenkov, I. V., Walter, M. C., Rattei, T. & Mewes, H.-W. Beyond the ‘best’ match: machine learning annotation of protein sequences by integration of different sources of information. *Bioinformatics* 24, 621–628 (2008).
- Titz, B. et al. The binary protein interactome of *Treponema pallidum* - The syphilis spirochete. *PLoS One* 3, e2292 (2008).
- Uetz, P. et al. A comprehensive analysis of protein-protein interactions in *Saccharomyces cerevisiae*. *Nature* 403, 623–627 (2000).
- Van den Bossche, A. et al. Systematic Identification of Hypothetical Bacteriophage Proteins Targeting Key Protein Complexes of *Pseudomonas aeruginosa*. *J. Proteome Res.* 13, 4446–4456 (2014).
- van Hal, S. J. et al. Predictors of Mortality in *Staphylococcus aureus* Bacteremia. *Clin. Microbiol. Rev.* 25, 362–386 (2012).
- Venkatesan, K. et al. An empirical framework for binary interactome mapping. *Nat. Methods* 6, 83–90 (2009).
- Vo, T. V. et al. A Proteome-wide Fission Yeast Interactome Reveals Network Evolution Principles from Yeasts to Human. *Cell* 164, 310–323 (2016).
- Wang, F. et al. *Mycobacterium tuberculosis* dihydrofolate reductase is not a target relevant to the antitubercular activity of isoniazid. *Antimicrob. Agents Chemother.* 54, 3776–82 (2010).
- Wang, G. & Maier, R. J. An NADPH quinone reductase of *Helicobacter pylori* plays an important role in oxidative stress resistance and host colonization. *Infect. Immun.* 72, 1391–6 (2004).
- Wang, J., Gao, Y. & Zhao, F. Phage-bacteria interaction network in human oral microbiome. *Environ. Microbiol.* (2015). doi:10.1111/1462-2920.12923

- Wang, M. et al. PaxDb, a Database of Protein Abundance Averages Across All Three Domains of Life. *Mol. Cell. Proteomics* 11, 492–500 (2012).
- Wang, P. I. & Marcotte, E. M. It's the machine that matters: Predicting gene function and phenotype from protein networks. *J. Proteomics* 73, 2277–2289 (2010).
- Wang, Y. et al. Global protein-protein interaction network in the human pathogen mycobacterium tuberculosis H37Rv. *J. Proteome Res.* 9, 6665–6677 (2010).
- Warnes GR, Bolker B, Bonebakker L, Gentleman R, Liaw WHA, L. T. gplots: Various R Programming Tools for Plotting Data. (2015).
- Washizaki, A., Yonesaki, T. & Otsuka, Y. Characterization of the interactions between Escherichia coli receptors, LPS and OmpC, and bacteriophage T4 long tail fibers. *Microbiologyopen* (2016). doi:10.1002/mbo3.384
- Waterhouse, A. M., Procter, J. B., Martin, D. M. A., Clamp, M. & Barton, G. J. Jalview Version 2--a multiple sequence alignment editor and analysis workbench. *Bioinformatics* 25, 1189–1191 (2009).
- Weigel, C. & Seitz, H. Bacteriophage replication modules. *FEMS Microbiol. Rev.* 30, (2006).
- Wiles, A. M. et al. Building and analyzing protein interactome networks by cross-species comparisons. *BMC Syst. Biol.* 4, 36 (2010).
- Wilson, D. N. & Nierhaus, K. H. Ribosomal Proteins in the Spotlight. *Crit. Rev. Biochem. Mol. Biol.* 40, 243–267 (2005).
- Wolf, D. H. & Hilt, W. The proteasome: a proteolytic nanomachine of cell regulation and waste disposal. *Biochim. Biophys. Acta - Mol. Cell Res.* 1695, 19–31 (2004).
- Wu, X. et al. FmvB: A Francisella tularensis Magnesium-Responsive Outer Membrane Protein that Plays a Role in Virulence. *PLoS One* 11, e0160977 (2016).
- Wuchty, S., Oltvai, Z. N. & Barabási, A.-L. Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nat. Genet.* 35, 176–179 (2003).
- Wuchty, S. & Uetz, P. Protein-protein Interaction Networks of E. coli and S. cerevisiae are similar. *Sci. Rep.* 4, 7187 (2014).

- Xu, P. et al. Genome-wide essential gene identification in *Streptococcus sanguinis*. *Sci. Rep.* (2011).
doi:10.1038/srep00125
- Yegambaram, K., Bulloch, E. M. M. & Kingston, R. L. Protein domain definition should allow for conditional disorder. *Protein Sci.* 22, 1502–1518 (2013).
- Young, K. H. Yeast two-hybrid: so many interactions, (in) so little time... *Biol. Reprod.* 58, 302–311 (1998).
- Yu, H. et al. High-quality binary protein interaction map of the yeast interactome network. *Science* 322, 104–110 (2008).
- Yus, E. et al. Impact of genome reduction on bacterial metabolism and its regulation. *Science* 326, 1263–1268 (2009).
- Zhang, M., Su, S., Bhatnagar, R. K., Hassett, D. J. & Lu, L. J. Prediction and Analysis of the Protein Interactome in *Pseudomonas aeruginosa* to Enable Network-Based Drug Target Selection. *PLoS One* 7, e41202 (2012).
- Zhang, P. et al. Isolation, subunit composition and interaction of the NDH-1 complexes from *Thermosynechococcus elongatus* BP-1. *Biochem. J.* 390, 513–520 (2005).
- Zhong, Q. et al. An inter-species protein-protein interaction network across vast evolutionary distance. *Mol. Syst. Biol.* 12, 865–865 (2016).
- Zipperer, A. et al. Human commensals producing a novel antibiotic impair pathogen colonization. *Nature* 535, 511–516 (2016).

VITA

John Harry Caufield was born on March 23, 1986 in Chester County, Pennsylvania. He graduated from Downingtown East High School, Lionville, Pennsylvania in 2004 and received a Bachelor of Science degree in Biological Sciences from the University of Delaware in 2008. He received a Master of Science degree in Bioinformatics from Virginia Commonwealth University in 2012.

APPENDIX I

Guide to *spicednog*

I.I. User's guide to *spicednog*

Spicednog (SPecific Conservation for Every DamN Orthologous Group) is a set of Python scripts intended for parsing and retrieving orthology assignments for a given set of genes or genomes. All scripts are intended to be run from the Linux command line. Orthology data is provided by the eggNOG project v.3 (http://eggnog.embl.de/version_3.0/). These scripts will not work properly with newer versions of eggNOG as the data file structure has changed. *Spicednog* is intended for use with bacterial genomes and bacterial gene orthology.

These scripts have been written for Python 2.7 and have not been fully tested with Python 3. They require an Internet connection.

There are three main components:

- *spicednog.py* takes a species or strain name and provides lists of genes, orthologous groups, and basic counts of locus types.
- *spicednog-convert.py* takes lists of Uniprot IDs and converts them into OG, NOG, and bactNOG IDs if available.
- *spicednog-marshmallow.py* takes an OG ID and finds genomes which contain it.

The accessory module *ConToComplexCon.py* is also provided to assist with calculating conservation fractions for protein complexes.

I.I.I Setup

All *spicednog* should be extracted to the same folder. This folder must also contain the eggNOG v.3 species flat file (species.v3.txt) as well as the contents of the compressed eggNOG members file (all.members.tar.gz; this file will decompress to a single folder, all.members) and the contents of the compressed eggNOG protein aliases file (protein.aliases.v3.txt.gz; this file decompresses to a single 3 Gb text file). Both files are available at http://eggnog.embl.de/version_3.0/downloads.html.

I.I.II Running *spicednog*

The main script can then be run by navigating to its location and running as shown in **Figure A1**. Here, python2 is specified to ensure the correct version is running, but may depend on the Python version installed on the system.

```
harry@Bilobe-mint ~/Documents/spicednog-master $ python2 spicednog.py
Which species are you looking for?
> 
```

Fig. A1. The initial *spicednog* prompt.

Providing a species name causes *spicednog* to search the reference proteome database within Uniprot for entries matching the search query. Results will resemble those shown in **Figure A2**. In the event of no match, the search should be repeated.

```
Which species are you looking for?
> Streptococcus pneumoniae
##taxon type      name_official      name_compact      name_NCBI      name_imported      nr_of_loci
488221 peripheral species Streptococcus pneumoniae 78585 Streptococcus pneumoniae 78585 Streptococcus pneumoniae 78585 Streptococcus p
neumoniae 78585 2282
561276 peripheral species Streptococcus pneumoniae ATCC 700669 Streptococcus pneumoniae 700669 Streptococcus pneumoniae ATCC 700669 S
treptococcus pneumoniae ATCC 700669 1998
516950 peripheral species Streptococcus pneumoniae CGSP14 Streptococcus pneumoniae CGSP14 Streptococcus pneumoniae CGSP14 Streptococcus p
neumoniae CGSP14 2206
373153 peripheral species Streptococcus pneumoniae D39 Streptococcus pneumoniae D39 Streptococcus pneumoniae D39 Streptococcus p
neumoniae D39 1914
512566 peripheral species Streptococcus pneumoniae G54 Streptococcus pneumoniae G54 Streptococcus pneumoniae G54 Streptococcus p
neumoniae G54 2115
487214 peripheral species Streptococcus pneumoniae Hungary19A 6 Streptococcus pneumoniae 19A6 Streptococcus pneumoniae Hungary19A-6 S
treptococcus pneumoniae Hungary19A 6 2155
488222 peripheral species Streptococcus pneumoniae JJA Streptococcus pneumoniae JJA Streptococcus pneumoniae JJA Streptococcus p
neumoniae JJA 2123
488223 peripheral species Streptococcus pneumoniae P1031 Streptococcus pneumoniae P1031 Streptococcus pneumoniae P1031 Streptococcus p
neumoniae P1031 2073
171101 peripheral species Streptococcus pneumoniae R6 Streptococcus pneumoniae R6 Streptococcus pneumoniae R6 Streptococcus p
neumoniae R6 2042
487213 peripheral species Streptococcus pneumoniae Taiwan19F 14 Streptococcus pneumoniae 19F14 Streptococcus pneumoniae Taiwan19F-14 S
treptococcus pneumoniae Taiwan19F 14 2044
170187 core species Streptococcus pneumoniae TIGR4 Streptococcus pneumoniae TIGR4 Streptococcus pneumoniae TIGR4 Streptococcus pneumoniae
TIGR4 2105
There were 11 matches. Your species code may be 170187.
If that is acceptable, enter Y. If not, type your chosen species code (the first number).
> 
```

Fig. A2. The results of a *spicednog* search.

Results contain the following information:

- **taxon**: the taxonomy ID specific to this entry. Used by Uniprot and the NCBI Taxonomy database.
- **type**: may be “peripheral species” or “core species”. Core species are more likely to be representative models.
- **name_official**: The full name of the corresponding species and strain.
- **name_compact**: A shorter form of the name.
- **name_NCBI**: The name used by NCBI databases.
- **nr_of_loci**: the total count of protein-coding loci in this species and strain's genome.

Spicednog provides a suggestion of the best match to the query based on which of the results is a core species, if any. Otherwise, the entry may be chosen by typing the corresponding **taxon** value.

Once a reference proteome is selected, *spicednog* generates four files in the current directory.

[species name] OGs.txt contains on each line:

- An eggNOG v.3 OG. This may be a COG, NOG, or bactNOG.
- A single protein ID from the corresponding reference proteome. This is the internal ID used by eggNOG.
- The count of proteins in this set matching this OG. (This value includes *all possible OG matches* and will therefore not match values in other files.)
- The total count of members of the OG.

[species name] OG counts.txt contains on each line:

- An OG found in the set.
- The number of proteins in the set matching this OG.

[species name] loci counts.txt contains on each line:

- A protein found in the set.
- The number of potential OG matches for this protein.

[species name] conservation across Bacteria.txt contains on each line:

- An OG found in the set.
- A count of *bacterial genomes* also containing this OG.

The *spicednog* output also includes descriptive statistics of the reference proteome in terms of how many proteins can be mapped to OGs of any type and how many are highest-level OGs (that is, COGs) vs. more specific NOGs.

I.I.III Running accessory scripts

Spicednog-convert.py takes a list of Uniprot protein IDs as input and returns eggNOG IDs. This is useful for determining which proteins may map to multiple OGs. These IDs must be provided in a separate file. The script then prompts the user for the filename. The script searches the alias file for a corresponding eggNOG protein/locus ID, COG, NOG, and bactNOG. Results of NA indicate a corresponding ID was not found. An example of *spicednog-convert.py* output is shown in **Figure A3**.

```

harry@Bilobe-mint ~/Documents/spicednog-master $ python2 spicednog-convert.py
Please enter Uniprot ID list file
> "these_ids"
upid      locus      cogID      nogID      bactnogID
P13801    224308.BSU15310 NA      NOG08135      bactNOG66183
P69739    316385.ECDH10B_1042 COG1740 NA      bactNOG00893
P01553    7227.FBpp0071989 NA      NOG26079      NA
P96131    243276.TP0362    COG0227 NA      bactNOG45091

```

Fig. A3. The results of an example *spicednog-convert.py* search.

Spicednog-marshmallow.py searches for a given OG among a given set of species. This list may be customized (e.g., to search only Proteobacteria or another taxonomic group) but must be in the same directory as *spicednog-marshmallow.py*, must be named "speclist.txt", and must contain one taxonomy ID per line. Note that some species or strains may not be present in all databases and may return OG counts of zero. The name of one or more OGs must be provided at the command line. The script then returns the number of times the OG is found in each of the given species with IDs provided in the species list. An example of *spicednog-marshmallow.py* output is shown in **Figure A4**.

```

harry@Bilobe-mint ~/Documents/spicednog-master $ python2 spicednog-marshmallow.py "COG1234"
spicednog-marshmallow.py
Searching for 1 OGs in total.
Loaded COG list.
Searching for COG1234...
*Species*      COG1234
224308         2
272563         1
243276         1
85963          0
83333          0

```

Fig. A4. The results of an example *spicednog-marshmallow.py* search.

I.II. Code

I.II.I *spicednog.py*

```
#!/usr/bin/python
# SpecIfic Conservation for Every DamN Orthologous Group - SPICEDNOG
# For parsing eggNOG files on a single-species or strain basis.
# Works properly when in same directory as the following:
#     species.v3.txt
#     Extracted "all.members.tar.gz"
# Optimized for bacteria.
#INPUT: name of desired species or eggNOG identifier. Can enter as command line argument or when prompted

import mmap, re, sys, string, os

# Define input files
filenameSpecies = "species.v3.txt"
filenameCOG = "all.members/COG.members.txt"
filenameNOG = "all.members/NOG.members.txt"
filenamebactNOG = "all.members/bactNOG.members.txt"

# Open the species file and prompt for species name
txt = open(filenameSpecies)
# print "The species file is %r:" % filenameSpecies
print "Which species are you looking for?"
if (len(sys.argv)>1):
    print str(sys.argv[1])
    speciesname = str(sys.argv[1])
else:
    speciesname = raw_input("> ")

# Search the species file for matching rows, get species code and display them
speciesmatches = 0
print txt.readline()
for line in txt:
    if re.search(speciesname, line):
        speciescode = re.match('[0-9]{4,}', line)
        if re.search("core species", line):
            bestspeciescode = speciescode
        speciesmatches = speciesmatches + 1
        print line,

# If there is one match, use that one. If there are >1 matches, allow the user to choose by species code
if speciesmatches != 0:
```



```

if speciesmatches == 1:
    print "Your species code is %r." % speciescode.group()
    speciescode = speciescode.group()
else:
    try:
        bestspeciescode
    except NameError:
        print "More than one entry was matched. Aborting this run."
        sys.exit(0)
    else:
        speciescode = bestspeciescode
    print "There were %s matches. Your species code may be %s." % (speciesmatches, speciescode.group())
    print "If that is acceptable, enter Y. If not, type your chosen species code (the first number)."
    if (len(sys.argv)>1):
        #For automation purposes. Otherwise will wait for input when 4-
digit codes used
        confirms = "Y"
    else:
        confirms = raw_input("> ")
    if confirms == "Y":
        speciescode = speciescode.group()
    else:
        speciescode = confirms
else:
    sys.exit("There were no matches. This is how things go sometimes.")

# Go back and get one species (row) entry in case a new one was specified
# Retrieve its name and total number of loci
txt.seek(0)
for line in txt:
    if re.match(speciescode + '\s', line):
        speciesFullName = re.search('[A-Z]{1}[a-z]+\s[a-z]+(\s\w+)?(\s\w+)?', line)
        speciesAllLoci = float((re.search('[0-9]+$', line)).group())
        print "Species name: %s. Number of loci: %s." % (speciesFullName.group(), '{:g}'.format(speciesAllLoci))

# Move on to the COG and NOG files
txt.close()
ogfile=open(speciesFullName.group() + ' OGs.txt', 'w+')
print "\nOK. Building lists..."

# Retrieve rows from OG files with the corresponding species code. Write to the same output file
# Also populate the list of bacterial species while we have that bactNOG file open
loci = 0
bactnogloci = 0
txt2 = open(filenameCOG)
for line in txt2:

```

```

        if re.search("\t" + speciescode + "\.\w+", line):
            #print line
            loci = loci + 1
            ogfile.write(line),
txt3 = open(filenameNOG)
for line in txt3:
    if re.search("\t" + speciescode + "\.\w+", line):
        #print line
        loci = loci + 1
        ogfile.write(line),
txt4 = open(filenamebactNOG)
listOfBacteria = [0]                #To be a bacterial species, a code must be used with at least one bactNOG.
for line in txt4:
    if (re.search('\t[0-9]{4,}\.', line)):
        #print (re.search('(?:\t)([0-9]{4,})(?:\.)', line)).group(1)
        listOfBacteria.append(re.search('(?:\t)([0-9]{4,})(?:\.)', line).group(1))
    if re.search("\t" + speciescode + "\.\w+", line):
        #print line
        bactnogloci = bactnogloci + 1
        ogfile.write(line),
setOfBacteria = (set(listOfBacteria))
if (bactnogloci>0):
    print "This is one of 943 bacterial species/strains in the database."
else:
    print "This is not a bacterial species."
    sys.exit(0)  #This is just here for automation purposes. Comment out when using non-bacteria
ogfile.seek(0)
#for line in ogfile:
    #print line
print "\nSee %s for the OG list." % (ogfile.name)

# Get the number of times each OG is present - a rough analog for paralogy. Duplicates are removed.
# print "Looking at OGs in %s" % (ogfile.name)

ogfileogcounts=open(speciesFullName.group() + ' OG counts.txt', 'w')
with ogfile as f:
    filesize = os.path.getsize(speciesFullName.group() + ' OGs.txt')
    data = mmap.mmap(f.fileno(), filesize)
    #while True:
        #lineline = data.readline()
        #if lineline == "": break
        #print lineline
    alloGlist = re.findall('[C|N]OG[0-9]{4,6}', data)
    alloGset = (set(alloGlist))
    if alloGset:

```

```

        for i in allOGset:
            countOG = len(re.findall(i, data))
            ogcountline = "\n" + i + "\t" + str(countOG)
            #print ogcountline
            ogfileogcounts.write(ogcountline),
print "\nSee %s for the OG counts." % (ogfileogcounts.name)
ogfileogcounts.close()

# Get the number of times each locus is present - shows which loci are in >1 OG.
# raw_input("\nPress Enter to see how many times each locus is present.")

ogfilelocicounts=open(speciesFullName.group() + ' loci counts.txt', 'w')
alllocilist = re.findall(speciescode + '\.\w+', data)
alllociset = (set(alllocilist))
if alllociset:
    for i in alllociset:
        countlocus = len(re.findall(i, data))
        locuscountline = "\n" + i + "\t" + str(countlocus)
        #print locuscountline
        ogfilelocicounts.write(locuscountline),
print "\nSee %s for the locus counts." % (ogfilelocicounts.name)
ogfilelocicounts.close()

print "\n* Within %s there are %s loci which map to OGs. %s loci are unique (that is, they don't share OGs)." %
(speciesFullName.group(), loci, len(alllociset))
if bactnogloci >1:
    print "* There are %s highest-level OGs and %s bactNOG loci." % ((len(allOGset) - bactnogloci), bactnogloci)
else:
    print "* There are %s OGs." % (len(allOGset))
print "* OG loci, including any level of NOGs, comprise %s of all loci for this entry. %s loci did not map to OGs." %
('{:.2%}'.format(len(alllociset) / speciesAllLoci), '{:g}'.format(speciesAllLoci - len(alllociset)))
print "In summary: /| %s | %s | %s | %s |/" % (speciesFullName.group(), speciescode, '{:g}'.format(speciesAllLoci),
len(alllociset))

#print "\nPress Enter to get OG conservation across the whole database,\n\ttype B to restrict the search to Bacteria,\n\tOR
type X to exit."
#confirms2 = raw_input("> ")
print "Moving on to the bacterial conservation search."
confirms2 = "B"
if confirms2 == "X":
    sys.exit("Bye!")

#If requested, open up the COG and NOG and bactNOG files, split by species and remove duplicates, then search for all OGs
found above
#If just looking at bacterial conservation, retrieves only OGs from bacterial species.

```

```

print "\nOK, searching. This may take a while."
if confirms2 == "B":
    ogfileAllConserve=open(speciesFullName.group() + ' conservation across Bacteria.txt', 'w')
else:
    ogfileAllConserve=open(speciesFullName.group() + ' conservation across the database.txt', 'w')
cogs = mmap.mmap(txt2.fileno(), 0, prot=mmap.PROT_READ)
cogs = re.split('\..+(\n|\Z)', cogs)
cogset = (set(cogs))
if confirms2 == "B":
    filteredcogset = set([])
    for i in cogset:
        if re.search('(?:\t)([0-9]{4,})', i):
            if re.search('(?:\t)([0-9]{4,})', i).group(1) in setOfBacteria:
                filteredcogset.add(i)
    cogset = filteredcogset
    print "Filtered COGs for bacteria only."
cogsettogether = '\t'.join(cogset)
nogs = mmap.mmap(txt3.fileno(), 0, prot=mmap.PROT_READ)
nogs = re.split('\..+(\n|\Z)', nogset)
nogset = (set(nogs))
if confirms2 == "B":
    filterednogset = set([])
    for i in nogset:
        if re.search('(?:\t)([0-9]{4,})', i):
            if re.search('(?:\t)([0-9]{4,})', i).group(1) in setOfBacteria:
                filterednogset.add(i)
    nogset = filterednogset
    print "Filtered NOGs for bacteria only."
nogsettogether = '\t'.join(nogset)
bactnogs = mmap.mmap(txt4.fileno(), 0, prot=mmap.PROT_READ)
bactnogs = re.split('\..+(\n|\Z)', bactnogs) #Saves time by not filtering bactNOGs - they're already just in bacteria
bactnogset = (set(bactnogs))
bactnogsettogether = '\t'.join(bactnogset)
allOGlistCon = re.findall('[a-z]*[A-Z]{3}[0-9]{4,6}', data) #Get all the OGs for our chosen species.
print "Got all the OGs for the target species."
#print allOGlistCon
allOGsetCon = (set(allOGlistCon))
#print allOGsetCon
if allOGsetCon:
    for i in allOGsetCon:
        countOG = 0
        if 'COG' in i:
            countOG = (len(re.findall(i + '\t[0-9]+', cogsettogether))) #Finds one instance of the OG per species
code.
    ogcountline = "\n" + i + "\t" + str(countOG)

```

```

        ogfileAllConserve.write(ogcountline),
    if 'bactNOG' in i:
        countOG = (len(re.findall(i + '\t[0-9]+', bactnogsettogether))) #Ditto.
        ogcountline = "\n" + i + "\t" + str(countOG)
        ogfileAllConserve.write(ogcountline),
    if 'NOG' in i and not 'bactNOG' in i:
        countOG = (len(re.findall(i + '\t[0-9]+', nogsettogether))) #Same here. Leaves out missing ones to avoid
double-counting bactNOGs
        ogcountline = "\n" + i + "\t" + str(countOG)
        ogfileAllConserve.write(ogcountline),
print "\nSee %s for the OG counts." % (ogfileAllConserve.name)
sys.exit(0)

```

I.II.II *spicednog-convert.py*

```

#!/usr/bin/python
# SPecIfic Conservation for Every DamN Orthologous Group - SPICEDNOG
# convert module - for turning Uniprot IDs into other things
# Works properly when in same directory as the following:
#     protein.aliases.v3.txt
#     Extracted "all.members.tar.gz"
# Optimized for bacteria.
#INPUT: A list of Uniprot IDs, one per line, in file
import mmap, re, sys, string, os

# Define input files
filenameAliases = "protein.aliases.v3.txt"
filenameCOG = "all.members/COG.members.txt"
filenameNOG = "all.members/NOG.members.txt"
filenamebactNOG = "all.members/bactNOG.members.txt"

# prompt for upids
if (len(sys.argv)>1):
    #print str(sys.argv[1])
    filenameupid = str(sys.argv[1])
else:
    print("Please enter Uniprot ID list file")
    filenameupid = input("> ")
idfile = open(filenameupid)
print("upid\tlocus\tcogID\tnogID\tbactnogID")

# Search the input file for matching rows.
#Just returns the first matching hit.
undefinedvar = 'undefined'

```

```

for line in idfile:
    upid = line.rstrip()
    #print("***NOW SEARCHING ALIAS FILE FOR LOCUS FOR " + upid + "****")
    txt = open(filenameAliases)
    for line in txt:
        #print line
        locusline = undefinedvar
        if upid in line:
            locusline = line
            #print line
            break
        if locusline is undefinedvar:
            locus = locusline
    locuslist=locusline.split("|")
    locus = locuslist[0]
    #print locus
# Move on to the COG and NOG files
txt.seek(0,0)

## Retrieve rows from OG files with the corresponding species code. Write to the same output file
cogID = "NA"
nogID = "NA"
bactnogID = "NA"
txt2 = open(filenameCOG)
#print("***NOW SEARCHING COG FILE FOR LOCUS FOR " + upid + "****")
for line in txt2:
    if locus is undefinedvar:
        break
    if locus in line:
        #print line
        cogID = line[0:7]
        break
txt2.close()
txt3 = open(filenameNOG)
#print("***NOW SEARCHING NOG FILE FOR LOCUS FOR " + upid + "****")
for line in txt3:
    if locus is undefinedvar:
        break
    if locus in line:
        #print line
        nogID = line[0:8]
        break
txt3.close()
txt4 = open(filenamebactNOG)
#print("***NOW SEARCHING bactNOG FILE FOR LOCUS FOR " + upid + "****")

```

```

    for line in txt4:
        if locus is undefinedvar:
            break
        if locus in line:
            #print line
            bactnogID = line[0:12]
            break
    txt4.close()
    print(upid + "\t" + locus + "\t" + cogID + "\t" + nogID + "\t" + bactnogID)
# Output everything, one line each ID

sys.exit(0)

```

1.II.III spicednog-marshmallow.py

```

#!/usr/bin/python
# SPecIfic Conservation for Every DamN Orthologous Group - SPICEDNOG
# OG presence helper
# Input: At the command line, the name of the set (usually a protein complex name; don't use spaces)
# followed by the name of one or more eggNOG OGs (i.e., COG1234).
#       COGs, NOGs, and bactNOGs will work.
# Output: A taxon ID number and the number of members of the specified OG its genome contains.

import sys, array

specieslist = open("speclist.txt") #This is just a list of NCBI taxon IDs (Also used by eggNOG), one on each line.
listOfSpec = []
listOfOG = []
searchOGs = []

if (len(sys.argv)>1):
    for eacharg in sys.argv:
        searchOGs.append(eacharg)
    #del searchOGs[0]
    groupname = searchOGs[0]
    del searchOGs[0]
    print groupname
    print "Searching for %s OGs in total." % len(searchOGs)

else:
    sys.exit("No OGs provided.")
resultsList = [0]

```

```

#Set up the arrays of species and OGs to search.
for line in specieslist:
    listOfSpec.append(line.rstrip())
specieslist.close()
if any("COG" in item for item in searchOGs):
    coglist = open("all.members/COG.members.txt")
    for line in coglist:
        listOfOG.append(line)
    print "Loaded COG list."
    coglist.close()
if any("bactNOG" in item for item in searchOGs):
    bactnoglist = open("all.members/bactNOG.members.txt")
    for line in bactnoglist:
        listOfOG.append(line)
    print "Loaded bactNOG list."
    bactnoglist.close()
if any("NOG" in item for item in searchOGs):
    noglist = open("all.members/NOG.members.txt")
    for line in noglist:
        listOfOG.append(line)
    print "Loaded NOG list."
    noglist.close()

for eachOG in searchOGs:
    print "Searching for %s..." % eachOG
    resultsLine = -1
    oneSpeciesResults = []
    for i in listOfSpec:
        resultsLine = resultsLine + 1
        positivecount = 0
        for jline in listOfOG:
            if i in jline and eachOG in jline:
                positivecount = positivecount + 1
        #print "%s\t%s" % (i, positivecount)
        oneSpeciesResults.append(positivecount)
        #print oneSpeciesResults
    resultsList.append(oneSpeciesResults)

print "*Species*\t%s" % '\t'.join(map(str, searchOGs))
index = 0
for item in listOfSpec:
    tempString = "%s\t" % item
    index2 = 1
    for OG in searchOGs:
        tempString = tempString + "\t" + str(resultsList[index2][index])

```



```

        index2 = index2 + 1
    print tempString
    index = index + 1
    #On the same line, print the corresponding results row

```

I.II.IV *ConToComplexCon.py*

```

#!/usr/bin/python
# Script for converting spicednog-marshmallow-simple output to fractional conservation for
# each complex in a set of complexes.
# INPUTS: A matrix of conservation per species, with one species per row and
# one OG per column. A second file contains a list of complexes and their OG components.
# OUTPUT: A matrix like the first input file, but with one complex per column.
# Output values are fractional conservation (that is, (# conserved vs. ref)/(# of components in ref. complex))

import sys, array

conlist = open("EcoCyc_and_Hu_complex_component_conservation.txt")
complexlist = open("Hu_E_coli_complexes.txt")
fractionsf = open('Hu_complex_conservation_fractions.txt', 'w')

#Load all the complexes, storing components in one list and names in another
#They should all remain in the same order though
allcomplexes = []
allnames = []
for line in complexlist:
    singlecomplex = (line.rstrip()).split("\t")
    singlename = singlecomplex.pop(0)
    allcomplexes.append(singlecomplex)
    allnames.append(singlename)
#print(allcomplexes)

#Load all components we have conservation for - list includes first column for consistency
possiblecomponents = ((conlist.readline()).rstrip()).split("\t")
print("Using conservation of " + str(len(possiblecomponents)) + " components in " + str(len(allnames)) + " complexes.")

#Check to make sure the data matches up
missingcount = 0
missingcomponents = []
for itercomplex in allcomplexes:
    for component in itercomplex:
        if component in possiblecomponents:
            continue

```

```

        #print(component + " is in the set.")
    else:
        #print(component + " is NOT IN THE SET.")
        missingcomponents.append(component)
        missingcount = missingcount + 1
if missingcount > 0:
    print(str(missingcount) + " components are missing in the conservation data.")
    print(missingcomponents)

#Set up the output file
allnames.insert(0,"Species")
fractionsf.write("\t".join(allnames) + "\n")

#Iterate through all species and complexes
errors = 0 #The number of components which couldn't be found in the set
for line in conlist:
    line = (line.rstrip()).split("\t")
    fractionsf.write(line[0] + "\t")
    for itercomplex in allcomplexes:
        totalcon = 0
        for component in itercomplex:
            try:
                whichone = possiblecomponents.index(component)
                totalcon = totalcon + int(line[whichone])
                #print(totalcon)
            except ValueError:
                errors = errors + 1
                #print("Can't find " + component + " in the search set!")
        fractioncon = (totalcon / float(len(itercomplex)))
        fractionsf.write(str(fractioncon) + "\t")
        #if fractioncon:
            #print("%s\t%s\t%s\t%s" % (line[0], itercomplex, totalcon, len(itercomplex)))
        #fractionsf.write(repr(fractioncon))
        #print(line[0] + "\t" + str(fractioncon))
    fractionsf.write("\n")
print("Wrote output to " + fractionsf.name)

```

APPENDIX II

Guide to *network_umbra*

II.I. User's guide to *network_umbra*

Network_umbra predicts interactions in a protein interaction network based off a meta-interactome network. It is intended for use with bacterial proteins and offers some support for viral proteins. This set of scripts is intended to be run from the Linux command line. It was written for Python 2.7 but should be compatible with Python 3.

Network_umbra uses eggNOG v.4 for orthology assignments (releases v.4.1 have been confirmed to work as expected, though previous versions will not). These scripts also provide options for retrieving reference proteomes from Uniprot, assigning their proteins to orthologs, and predicting interactions among those orthologs.

II.I.I Setup

Network_umbra requires at least 5 GB of free disk space for input and output files. It will download the required files if they are not found and therefore requires an Internet connection for this purpose.

It also requires the following:

- **Biopython 1.65** or more recent. Try installing with the **pip** package installer as follows:

```
pip install numpy
```

```
pip install biopython
```

or see <http://biopython.org/DIST/docs/install/Installation.html>

- **BeautifulSoup 4**. Install as follows:

```
pip install beautifulsoup4
```

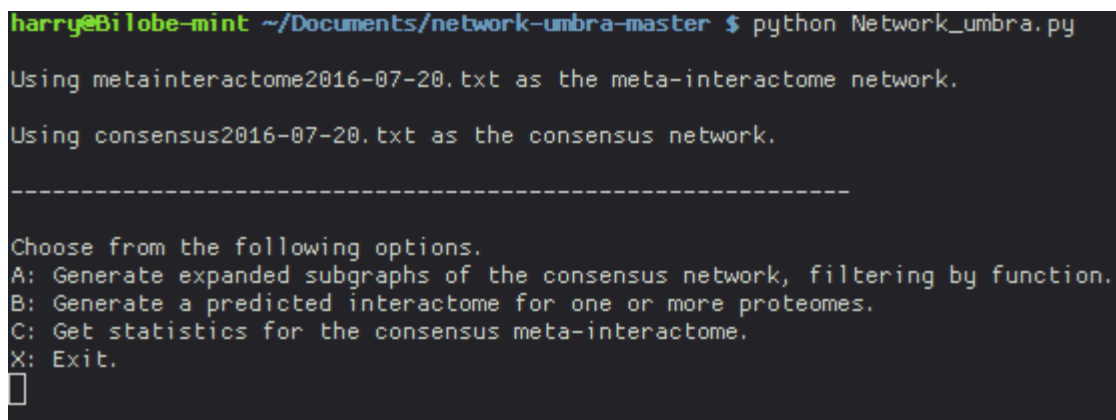
or see <http://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Network_umbra provides the option to download all available protein-protein interactions for bacteria from the IntAct database. Due to differences in data availability, interactions retrieved in this way may not directly correspond to those available through the IntAct HTML interface. See the EBI PSIQUIC View page (<http://www.ebi.ac.uk/Tools/webservices/psicquic/view/main.xhtml>) for the status of source interaction databases. Alternatively, download interaction sets in PSI-MI TAB 2.7 format or convert other data files to this format. These input files should contain *no header row*.

II.I.II Running *Network-umbra*

The first time it is run, *network-umbra* will download several files from eggNOG, including a protein ID conversion file, OG membership files, and annotation files.

It will then assemble a meta-interactome network from the provided interaction data file(s). All interactions involving interactors other than proteins are removed before addition to the meta-interactome. Finally, the meta-interactome is compressed into a consensus meta-interactome. The user is then presented with an options menu (**Figure A5**).

A terminal window showing the output of running 'python Network_umbra.py'. The prompt is 'harry@Bilobe-mint ~/Documents/network-umbra-master \$'. The output shows two status messages: 'Using metainteractome2016-07-20.txt as the meta-interactome network.' and 'Using consensus2016-07-20.txt as the consensus network.' followed by a separator line of dashes. Then, it presents a menu: 'Choose from the following options.', 'A: Generate expanded subgraphs of the consensus network, filtering by function.', 'B: Generate a predicted interactome for one or more proteomes.', 'C: Get statistics for the consensus meta-interactome.', 'X: Exit.', and a cursor pointing to the first option.

```
harry@Bilobe-mint ~/Documents/network-umbra-master $ python Network_umbra.py
Using metainteractome2016-07-20.txt as the meta-interactome network.
Using consensus2016-07-20.txt as the consensus network.
-----
Choose from the following options.
A: Generate expanded subgraphs of the consensus network, filtering by function.
B: Generate a predicted interactome for one or more proteomes.
C: Get statistics for the consensus meta-interactome.
X: Exit.
█
```

Fig. A5. The *network-umbra* menu seen after meta-interactome construction.

The following output files will be created, along with corresponding folders in the current working directory:

- 'metainteractome[date].txt'

A meta-interactome composed of all available bacterial protein-protein interactions. Follows PSI-MI Tab27 format, with the addition of two ortholog identifiers per row. See format description at <https://code.google.com/p/psimi/wiki/PsimiTab27Format>

- 'meta_statistics[date].txt'

Contains statistics relevant to the produced meta-interactome.

- 'taxid_context[date].txt'

Contains NCBI taxonomy IDs, names, parent IDs, and domains for all input interactions. All domains should be Bacteria. Used as a reference if meta-interactome and consensus meta-interactome not built during the same session.

- 'consensus[date].txt'

A consensus meta-interactome composed of all available bacterial protein-protein interactions. This set of interactions compresses all unique proteins into their corresponding orthologous groups. Data in each column is the following, from left to right:

1. **InteractorA** The first interactor. Usually an OG.
2. **InteractorB** The second interactor. Usually an OG.
3. **InteractionCount** Count of individual PROTEIN interactions contributing to this consensus interaction, as per the meta-interactome.
4. **TaxonCount** Count of different taxons (here, a proxy for species) corresponding to the interaction. *Similar taxons are grouped together where possible*, e.g. two different E. coli K-12 strains considered E. coli K-12 only.
5. **Taxons** The taxons corresponding to this interaction.
6. **FuncCatA** Functional category of the first interactor
7. **DescA** Description of the first interactor
8. **FuncCatB** Functional category of the second interactor
9. **DescB** Description of the second interactor

- 'cons_statistics[date].txt'

Contains statistics relevant to the produced consensus meta-interactome.

Option A, the expanded subgraph filtering, filters the consensus meta-interactome by functional category and returns interactions in which one interactor is an OG and the other is an individual protein. These interactions can be filtered further by specifying how many different species/taxids each OG vs. OG interaction must have been observed in.

- 'subgraph_expansion__[FuncCat]__[date].txt'

A set of subgraphs of the consensus meta-interactome, filtered by conservation of interactions and function of interactors. Each line is one interaction between a consensus interactor and a unique protein, accompanied by the source taxid of the unique protein.

- 'subgraph_expansion__[FuncCat]nodes[date].txt'

Annotation file for the nodes in the expanded subgraphs. Each line is a single node and includes an OG or protein ID, a functional category if available, an OG description, and a protein description (each line will have one of the two types of description; protein descriptions are taken directly from the meta-interactome file).

Option B permits retrieval of one or more reference proteomes from Uniprot (see **Figure A6**). These proteomes are then used for interactome prediction. A warning is shown if a given proteome cannot be mapped to OGs or has limited mapping. All mapped proteomes are stored for later use.

```

Get a proteome from Uniprot? (Y/N)
y
Please specify a full or partial species name.
Leuconostoc kimchii
Result Accession Name
0 UP000002362 Leuconostoc kimchii (strain IMSNU 11154 / KCTC 2386 / IH
25) 762051 2128
Please choose a search result.
0
Retrieving proteome for Leuconostoc kimchii (strain IMSNU 11154 / KCTC 2386 / IH
25)
File written to proteome_raw_762051.txt
Get a proteome from Uniprot? (Y/N)
n
Will now map proteomes to OGs.
Setting up protein to OG maps.
Mapping proteins in proteome_raw_762051.txt
proteome_raw_762051.txt contains 2128 proteins. 1624 map to OGs.

Available proteome maps:
Leuconostoc kimchii IMSNU 11154 proteome_map_762051.txt
Predict interactomes for all above? (Y/N)

```

Fig. A6. Using *network-umbra* to retrieve a proteome.

Reference proteome retrieval may also be performed without constructing a meta-interactome first by running *proteins_umbra.py*.

Network_umbra then provides the option to predict interactomes for all OG-mapped proteomes (see **Figure A7**). It first checks for experimental interactions (specifically, interactions involving proteins directly from the species of interest, including spoke-model interactions) and then makes additional interaction predictions based on the presence of proteins with at least one interaction observed between corresponding OGs in a different species (that is, an interolog). The result is provided in the folder “predicted_interactomes” with the filename “pred_interactome[taxid].txt”. Each line in this file is a single interaction including the two protein interactors, two corresponding OGs, and whether the interaction is classified as Predicted or Experimental. A table of summary statistics of all interactomes predicted in the current batch is also created in the current working directory.

```

Available proteome maps:
Leuconostoc kimchii IMSNU 11154      proteome_map_762051.txt
Predict interactomes for all above? (Y/N)
y
Predicting interactomes for the above species.
Loading meta-interactome files.

Predicting interactome for proteome_map_762051.txt.
Checking for experimental interactions.
0
Making interaction predictions.
.....100.....200.....300.....400.....500.....600..
.....700.....800.....900.....1000.....1100.....1200.
.....1300.....1400.....1500.....1600.....1700.....1
800.....1900.....2000.....2100.....2200.....2300.....
...2400.....2500.....2600.....2700.....2800.....2900...
.....3000.....3100.....3200.....3300.....3400.....350
0.....3600.....3700.....3800.....3900.....4000.....
.4100.....4200.....4300.....4400.....4500.....4600.....
....4700.....4800.....4900.....5000.....5100.....5200.
.....5300.....5400.....5500.....5600.....5700.....5
800.....5900.....6000.....6100.....6200.....6300.....
...6400.....6500.....6600.....6700.....6800.....6900...
.....7000.....7100.....7200.7217
Found 0 experimental interactions (including spoke expansion) and made 7217 inte
raction predictions for Leuconostoc kimchii IMSNU 11154.

```

Fig. A7. Example of interactome prediction process.

Option C provides counts of consensus meta-interactome properties. These include unique interactors (including both OGs and unmapped proteins, treated as single-member OGs), interactions, and unique taxids.

II.II. Code

II.II.I *Network_umbra.py*

```
#!/usr/bin/python
#Network_umbra.py
'''
Predicts interactions in a protein interaction network based off a meta-interactome network.
Uses eggNOG v.4.1.
Written for Python 2.7. Not tested with Python 3.

REQUIRES: Biopython 1.65 or more recent
           Also needs at least 5 GB of available disk space to accomodate data files and output
           More space may be necessary for proteome files.

INPUT: Downloads all available protein-protein interactions for bacteria from IntAct.
        Alternatively, uses a provided PPI data file in PSI-MI TAB 2.7 format.
        REMOVE THE HEADER ROW if it's present!
        Downloads highest-level (LUCA) and bacteria-specific Uniprot ID to NOG mappings from eggNOG v.4.1.
        Downloads highest-level (LUCA) bacteria-specific NOG annotations from eggNOG v.4.1.

OUTPUT:
'metainteractome[date].txt'
    A meta-interactome composed of all available bacterial protein-protein interactions.
    Follows PSI-MI Tab27 format, with the addition of two ortholog identifiers per row.
    See format description at https://code.google.com/p/psimi/wiki/PsimiTab27Format

'meta_statistics[date].txt'
    Contains statistics relevant to the produced meta-interactome.

'taxid_context[date].txt'
    Contains NCBI taxonomy IDs, names, parent IDs, and domains for all input interactions.
    All domains should be Bacteria.
    Used as a reference if meta-interactome and consensus meta-interactome not
    built during the same session.

'consensus[date].txt'
    A consensus meta-interactome composed of all available bacterial protein-protein interactions.
    This set of interactions compresses all unique proteins into their corresponding orthologous groups.
```

Data in each column is the following, from left to right:

InteractorA The first interactor. Usually an OG.

InteractorB The second interactor. Usually an OG.

InteractionCount Count of individual PROTEIN interactions contributing to this consensus interaction, as per the meta-interactome.

TaxonCount Count of different taxons (here, a proxy for species) corresponding to the interaction.
Similar taxons have been grouped together where possible, e.g. two different E. coli K-12 strains are just considered E. coli K-12.

Taxons The taxons corresponding to this interaction.

FuncCatA Functional category of the first interactor

DescA Description of the first interactor

FuncCatB Functional category of the second interactor

DescB Description of the second interactor

'cons_statistics[date].txt'

Contains statistics relevant to the produced consensus meta-interactome.

'subgraph_expansion_[FuncCat]_[date].txt'

A set of subgraphs of the consensus meta-interactome, filtered by conservation of interactions and function of interactors.

Each line is one interaction between a consensus interactor and a unique protein, accompanied by the source taxid of the unique protein.

'subgraph_expansion_[FuncCat]_nodes_[date].txt'

Annotation file for the nodes in the expanded subgraphs.

'interactome_statistics_[date].txt'

Counts of interactors - proteins and OGs - participating in predicted interactomes.
Contains the following counts per input proteome:
Name, taxid, Proteins, ProteinsNotInPPI, ProteinsWithExpPPI, ProteinsWithPredPPI,
UniqueOGs, OGsWithoutInteractions, OGsWithExpInt, OGsWithPredInt, ExpOGIntNet, OGIntInPredNet

Uses PSIQUIC service to retrieve IntAct data - see <https://github.com/micommunity/psicquic>

...

import proteins_umbra

```

import glob, gzip, operator, os, re, requests, sys, urllib2, zipfile
from Bio import Entrez
from bs4 import BeautifulSoup
from collections import Counter
from datetime import date

Entrez.email = 'caufieldjh@vcu.edu'

#Options
useViruses = True          #Option for using eggNOG's viral OGs. Requires the filters permitting only Bacteria to be modified
                             #Also requires the viral OGs to be downloaded and added.
                             #This option needs to be set True BEFORE the Uniprot to OG map is built or it won't
include proteins from viruses

useNonRefProteomes = True    #Option to search non-reference Uniprot proteomes in the interactome prediction module
#Retrieving non-reference proteomes sometimes returns an empty response.
#This happens with proteomes only in UniParc (e.g., if they are redundant)
#In those cases, we reject the search result.

#Functions

def get_eggnoG_maps():
    #Download and unzip the eggNOG ID conversion file
    #Filters file to just Uniprot IDs; the resulting file is the map file.
    #One Uniprot ID may correspond to multiple OGs - e.g. COG1234,COG3810,COG9313.
    #these cases are considered OGs in their own right as this may indicate a pattern of conserved sequences on its own
    baseURL = "http://eggnogdb.embl.de/download/eggnoG_4.1/"
    convfilename = "eggnoG4.protein_id_conversion.tsv.gz"    #File contains ALL database identifiers and corresponding
    proteins

    convfilepath = baseURL + convfilename
    outfilepath = convfilename[0:-3]
    dl_convfile = 1 #If 1, we need to download
    if os.path.isfile(convfilename): #Already have the compressed file, don't download
        print("Found compressed ID conversion file on disk: %s" % convfilename)
        decompress_convfile = 1
        dl_convfile = 0
    if os.path.isfile(outfilepath): #Already have the decompressed file don't download
        print("Found ID conversion file on disk: %s" % outfilepath)

```

```

    decompress_convfile = 0
    dl_convfile = 0

if dl_convfile == 1:
    print("Downloading ID mapping file - this file is ~400 Mb compressed so this may take some time.")
    print("Downloading from %s" % convfilepath)
    response = urllib2.urlopen(convfilepath)
    compressed_file = open(os.path.basename(convfilename), "w+b") #Start local compressed file
    chunk = 1048576
    while 1:
        data = (response.read(chunk)) #Read one Mb at a time
        compressed_file.write(data)
        if not data:
            print("\n%s file download complete." % convfilename)
            compressed_file.close()
            break
        sys.stdout.flush()
        sys.stdout.write(".")
    decompress_convfile = 1

if decompress_convfile == 1:
    print("Decompressing map file. Lines written, in millions:")
    #Done in chunks since it's a large file
    with gzip.open(convfilename) as infile: #Open that compressed file, read and write to uncompressed file
        outfile = open(outfilepath, "w+b")
        linecount = 0
        for line in infile:
            outfile.write(line)
            linecount = linecount + 1
            if linecount % 100000 == 0:
                sys.stdout.write(".")
            if linecount % 1000000 == 0:
                sys.stdout.flush()
                sys.stdout.write(str(linecount/1000000))

        infile.close()
    newconvfilename = outfilepath
    outfile.close()

#Download and decompress member NOG files (2 of them)

```

```

nogURL = baseURL + "data/NOG/"
nogfilename = "NOG.members.tsv.gz"
bactnogURL = baseURL + "data/bactNOG/"
bactnogfilename = "bactNOG.members.tsv.gz"
all_nog_locations = [[nogURL, nogfilename], [bactnogURL, bactnogfilename]]

if useViruses == True:
    virnogURL = baseURL + "data/viruses/Viruses/"
    virnogfilename = "Viruses.members.tsv.gz"
    all_nog_locations.append([virnogURL, virnogfilename])

for location in all_nog_locations:
    baseURL = location[0]
    memberfilename = location[1]
    memberfilepath = baseURL + memberfilename
    outfilepath = memberfilename[0:-3]
    if os.path.isfile(memberfilename):
        print("\nFound compressed NOG membership file on disk: %s" % memberfilename)
        decompress_memberfile = 1
    if os.path.isfile(outfilepath):
        print("\nFound NOG membership file on disk: %s" % outfilepath)
        decompress_memberfile = 0
    else:
        print("\nDownloading NOG membership file - this may take some time.")
        print("Downloading from %s" % memberfilepath)
        response = urllib2.urlopen(memberfilepath)
        compressed_file = open(os.path.basename(memberfilename), "w+b") #Start local compressed file
        chunk = 1048576
        while 1:
            data = (response.read(chunk)) #Read one Mb at a time
            compressed_file.write(data)
            if not data:
                print("\n%s file download complete." % memberfilename)
                compressed_file.close()
                break
            sys.stdout.flush()
            sys.stdout.write(".")
        decompress_memberfile = 1

```

```

if decompress_memberfile == 1:
    print("Decompressing NOG membership file %s" % memberfilename)
    #Done in chunks since it's a large file
    with gzip.open(memberfilename) as infile: #Open that compressed file, read and write to uncompressed
file
        outfile = open(outfilepath, "w+b")
        linecount = 0
        for line in infile:
            outfile.write(line)
            linecount = linecount + 1
            if linecount % 100000 == 0:
                sys.stdout.write(".")
            if linecount % 1000000 == 0:
                sys.stdout.flush()
                sys.stdout.write(str(linecount/1000000))
        infile.close()
    outfile.close()

#Clean up by removing compressed files
print("\nRemoving compressed files.")
all_compressed_files = [convfilename, nogfilename, bactnogfilename]
if useViruses == True:
    all_compressed_files.append(virnogfilename)
for filename in all_compressed_files:
    if os.path.isfile(filename):
        os.remove(filename)

#Load and filter the ID conversion file as dictionary
print("Parsing ID conversion file. Lines read, in millions:")
with open(convfilename[0:-3]) as infile:
    id_dict = {} #Dictionary of eggNOG protein IDs with database IDs as keys
    #Gets filtered down to relevant database IDs (i.e., Uniprot IDs)
    linecount = 0
    for line in infile:
        linecount = linecount + 1
        line_raw = ((line.rstrip()).split("\t")) #Protein IDs are split for some reason; merge them
        one_id_set = [line_raw[0] + "." + line_raw[1], line_raw[2], line_raw[3]]
        if "UniProt_AC" in one_id_set[2]:
            id_dict[one_id_set[1]] = one_id_set[0]

```

```

        if linecount % 100000 == 0:
            sys.stdout.write(".")
        if linecount % 1000000 == 0:
            sys.stdout.flush()
            sys.stdout.write(str(linecount/1000000))

infile.close()

#Use filtered ID conversion input to map to NOG members
print("\nReading NOG membership files.")
all_nog_filenames = [nogfilename[0:-3], bactnogfilename[0:-3]]
nog_members = {}          #Dictionary of NOG ids with protein IDs as keys (need to split entries for each)
nog_count = 0
for filename in all_nog_filenames:
    temp_nog_members = {}  #We will have duplicates within each set but don't want to lose the information.
    print("Reading from %s" % filename)
    with open(filename) as infile:
        for line in infile:
            nog_count = nog_count + 1
            line_raw = ((line.rstrip()).split("\t"))
            nog_id = line_raw[1]
            line_members = line_raw[5].split(",")
            for protein_id in line_members:
                #The same protein could be in more than one
                if protein_id in temp_nog_members:
                    temp_nog_members[protein_id] = temp_nog_members[protein_id] + "," + nog_id
                else:
                    temp_nog_members[protein_id] = nog_id

    infile.close()
    nog_members.update(temp_nog_members)

upids_length = str(len(id_dict))
nogs_length = str(nog_count)
proteins_length = str(len(nog_members))

print("Mapping %s Uniprot IDs to %s NOGs through %s eggNOG protein IDs:" % (upids_length, nogs_length,
proteins_length))
upid_to_NOG = {}          #Conversion dictionary. Values are OGs, keys are UPIDs.
mapped_count = 0          #upids mapped to nogs.
for upid in id_dict:

```

```

        if id_dict[upid] in nog_members:
            upid_to_NOG[upid] = nog_members[id_dict[upid]]
            mapped_count = mapped_count + 1
            if mapped_count % 100000 == 0:
                sys.stdout.write(".")
            if mapped_count % 1000000 == 0:
                sys.stdout.flush()
                sys.stdout.write(str(mapped_count/1000000))

#Use this mapping to build map file, named "uniprot_og_maps_*.txt"
print("Writing map file.")
nowstring = (date.today()).isoformat()
mapfilename = "uniprot_og_maps_" + nowstring + ".txt"
mapfile = open(mapfilename, "w+b")
for mapping in upid_to_NOG:
    mapfile.write(mapping + "\t" + upid_to_NOG[mapping] + "\n")    #Each line is a uniprot ID and an OG id
mapfile.close()

def get_interactions():
    #Download and unzip the most recent IntAct version, filtered for bacteria, using REST
    #Just uses IntAct for consistency, but could theoretically include other PSICQUIC compatible DB's
    #May need to add more interactions to the file if not present in IntAct
    #The PSICQUIC interface may also not retrieve all available interactions or may not filter as desired,
    #so script prompts for option to use other input files.
    #See format description here: https://code.google.com/p/psimi/wiki/PsimiTab27Format

    #As of Feb 8 2016, this only downloads a few entries - seem to be an issue with PSICQUIC or IntAct or both.

    baseURL = "http://www.ebi.ac.uk/Tools/webservices/psicquic/intact/webservices/current/search/query/species:
%22taxid:2%22?format=tab27"
    intfilename = "protein-interactions.tab"

    if os.path.isfile(intfilename):
        print("Found default interaction file on disk: %s" % intfilename)
    else:
        response = urllib2.urlopen(baseURL)
        print("Downloading from IntAct. NOTE: This option may only provide enough interactions for an example.")
        intfile = open(os.path.basename(intfilename), "w+b") #Start local file
        chunk = 1048576

```



```

while 1:
    data = (response.read(chunk)) #Read one Mb at a time
    intfile.write(data)
    if not data:
        print("\nInteraction file download complete.")
        intfile.close()
        break
    sys.stdout.flush()
    sys.stdout.write(".")

def get_eggnog_annotations():
    #Downloads and extracts the eggNOG NOG annotations.
    baseURLs = ["http://eggnogdb.embl.de/download/latest/data/bactNOG/",
"http://eggnogdb.embl.de/download/latest/data/NOG/"]
    bactannfilename = "bactNOG.annotations.tsv.gz" #The annotations for bacteria-specific NOGs
    lucaannfilename = "NOG.annotations.tsv.gz" #The annotations for other NOGs, but not bacteria-specific NOGs
    annfilenames = [bactannfilename, lucaannfilename]

    if useViruses == True:
        baseURLs.append("http://eggnogdb.embl.de/download/latest/data/viruses/Viruses/")
        annfilenames.append("Viruses.annotations.tsv.gz")

    this_url = 0
    for annfilename in annfilenames:
        annfilepath = baseURLs[this_url] + annfilename
        this_url = this_url + 1
        outfilepath = annfilename[0:-3]
        if os.path.isfile(annfilename):
            print("Found compressed annotation file on disk: " + annfilename)
        else:
            response = urllib2.urlopen(annfilepath)
            print("Downloading from " + annfilepath)
            compressed_file = open(os.path.basename(annfilename), "w+b") #Start local compressed file
            chunk = 1048576
            while 1:
                data = (response.read(chunk)) #Read one Mb at a time
                compressed_file.write(data)
                if not data:
                    print("\n" + annfilename + " file download complete.")

```

```

        compressed_file.close()
        break
    sys.stdout.flush()
    sys.stdout.write(".")

    print("Decompressing annotation file.")
    with gzip.open(annfilename) as infile: #Open that compressed file, read and write to uncompressed file
        file_content = infile.read()
        outfile = open(outfilepath, "w+b")
        outfile.write(file_content)
        infile.close()
    outfile.close()

    print("\nRemoving compressed files.")
    all_compressed_files = [bactannfilename, lucaannfilename]
    for filename in all_compressed_files:
        os.remove(filename)

def build_meta(mapping_file_list, ppi_data):
    #Sets up the meta-interactome network.
    #Also creates statistics file about the meta-interactome.
    #This means unique proteins become referred to by their OGs.
    #Interactions are still unique, so two OGs may interact multiple times.

    nowstring = (date.today()).isoformat()
    meta_network_filename = "metainteractome" + nowstring + ".txt"
    taxid_context_filename = "taxid_context" + nowstring + ".txt"
    meta_network_file = open(meta_network_filename, "w")
    taxid_context_file = open(taxid_context_filename, "w")

    all_taxids = []
    all_filtered_taxids = [] #Will remove non-bacterial taxids, unless useViruses is on
    map_dict = {} #Uniprot ID to OG dictionary for ALL IDs
    interaction_file = open(ppi_data)
    interaction_array = []
    interaction_filtered_array = []
    protein_array = []
    taxid_species = {} #Dictionary to store taxids and their name and PARENT taxid.

```

```

print("Building meta-interactome...")
print("Setting up protein to OG maps.")
#We preferentially use bacteria mapping first
for input_map_file in mapping_file_list:
    try:
        map_file = open(input_map_file)
    except IOError as e:
        print("I/O error({0}): {1}".format(e.errno, e.strerror))
    for line in map_file:
        one_map = ((line.rstrip()).split("\t"))
        map_dict[one_map[0]] = one_map[1]
    map_file.close()

#for item in map_dict:
#    print(item + "\t" + map_dict[item])

print("Arraying interaction file and creating lists of proteins and taxids.")

for line in interaction_file:
    one_interaction = (line.rstrip()).split("\t")
    for taxid in [one_interaction[9], one_interaction[10]]:
        taxid = (((taxid.split("|"))[0]).lstrip("taxid:")).split("(")[0]
        if taxid not in all_taxids and taxid != "-": #Need to start filtering taxids here so we don't pass
bad values to Entrez
            #Why would we get this value anyway? Could be malformed entry as all interactions should have
taxids
            all_taxids.append(taxid) #Just the raw taxid list
            interaction_array.append(one_interaction) #This is just the raw interaction list at this point

interaction_file.close()

print("Finding details for interactor taxids. This will take some time.")

for taxid in all_taxids:
    if taxid in ["Taxid interactor A", "Taxid interactor B"]: #This means the header wasn't removed.
        continue

```

```

unique_taxid_count = 0
#print(str(taxid))
target_handle = Entrez.efetch(db="Taxonomy", id=str(taxid), retmode="xml")
target_records = Entrez.read(target_handle)
taxid_name = target_records[0]["ScientificName"]
taxid_parent = target_records[0]["ParentTaxId"]
taxid_division = target_records[0]["Division"]
#print(taxid_division)
taxid_filter = ["Bacteria"]
if useViruses == True:
    virus_types = ["Phages", "Viruses"]
    for virus_type in virus_types:
        taxid_filter.append(virus_type)
if taxid_division in taxid_filter: #Restrict the set to bacteria, unless useViruses is on
    taxid_species[taxid] = [taxid_name, taxid_parent, taxid_division]
    taxid_context_file.write(str(taxid) + "\t" + "\t".join(taxid_species[taxid]) + "\n")
    if taxid not in all_filtered_taxids:
        all_filtered_taxids.append(taxid)
        sys.stdout.flush()
        sys.stdout.write(".")
        unique_taxid_count = unique_taxid_count + 1
        if unique_taxid_count % 100 == 0:
            sys.stdout.flush()
            sys.stdout.write(str(unique_taxid_count))
        #print(taxid_species[taxid])
taxid_context_file.close()

if useViruses == False:
    print("\nCleaning up data by removing non-protein and non-bacterial interactors.")
else:
    print("\nCleaning up data by removing non-protein and non-bacterial and non-viral interactors.")
interactions_removed = 0
for interaction in interaction_array:
    interaction_ok = 1
    for taxid in [interaction[9], interaction[10]]:
        taxid = (((taxid.split("|"))[0]).lstrip("taxid:")).split("(")[0]
        if taxid not in all_filtered_taxids: #This is where the non-bacterial (and non-viral, if
useViruses is on) interactions get removed
            interaction_ok = 0 #Ensure the interaction won't be kept later

```

```

        break #Ignore this interactor and the other in the pair
    if interaction_ok == 1: #Don't bother to filter proteins if this didn't pass the first filter
        for protein in interaction[0:2]: #both proteins in the interacting pair
            if protein[0:9] != "uniprotkb": #Only keep proteins with uniprot IDs
                #Might be nice to keep other protein IDs too but they're rare
                interaction_ok = 0 #Ensure the interaction won't be kept later
                break #Ignore this interactor and the other in the pair
            this_protein = protein.lstrip("uniprotkb:")
            if this_protein not in protein_array:
                protein_array.append(this_protein)
        if interaction_ok == 1: # The last filter check for cleaning
            if interaction not in interaction_filtered_array:
                interaction_filtered_array.append(interaction)
        else:
            interactions_removed = interactions_removed + 1
    else:
        interactions_removed = interactions_removed + 1

print("Total taxids: %s" % (len(all_filtered_taxids)))
print("Total raw interactions: %s" % (len(interaction_array)))
print("Interactions removed: %s" % (interactions_removed))

print("Mapping OGs to %s proteins in %s interactions." % (len(protein_array), len(interaction_filtered_array)))

protein_og_maps = {} #Dictionary to save protein-OG mapping specific for this interaction set
mapped_count = 0
proteins_without_og = 0
for protein in protein_array:
    mapped_count = mapped_count + 1

    if protein in map_dict:
        matching_og = map_dict[protein]
    else:
        matching_og = protein #If the protein doesn't map to an OG it retains its original ID
        proteins_without_og = proteins_without_og + 1

    protein_og_maps[protein] = matching_og

```

```

print("\nWriting meta-interactome file.")
interaction_count = 0
for interaction in interaction_filtered_array:          #Write OGs for all (filtered) interactions.
    interaction_count = interaction_count +1

    for protein in interaction[0:2]:                  #Get matching OGs for both proteins in the pair.
        matching_OG = protein_OG_maps[protein.lstrip("uniprotkb:")]
        interaction.append(matching_OG)

    interaction_out = "\t".join(interaction) + "\n"
    meta_network_file.write(interaction_out)

    #print("mapped " + str(interaction_count) + " - " + matching_OG_A + " vs. " + matching_OG_B)

meta_network_file.close()

meta_stats_filename = "meta_statistics_" + nowstring + ".txt"
meta_stats_file = open(meta_stats_filename, "w")
stats_header = ("Unique proteins\tInteractions\tProteins without OG\n")
meta_stats_file.write(stats_header)
meta_statistics = []
for meta_stat in [len(protein_array), interaction_count, proteins_without_OG]:
    meta_statistics.append(str(meta_stat))
meta_stats_file.write("\t".join(meta_statistics))
print("\nWrote meta-interactome statistics to %s" % meta_stats_filename)
meta_stats_file.close()

return [meta_network_filename, taxid_species]

def build_consensus(metafile, annotation_file_list, taxid_species):
    #Sets up the consensus meta-interactome network.
    #This is identical to the meta-interactome but compresses interactions into their respective OGs.
    #Interactors without OG assignment are retained and considered single-member OGs.

    nowstring = (date.today()).isoformat()
    consensus_network_filename = "consensus" + nowstring + ".txt"
    consensus_network_file = open(consensus_network_filename, "w")

    consensus_interactors = []          #Consensus interactome interactors - an OG where mapping is possible

```

```

all_interactions_taxids = [] #All interactions in the meta-interactome, but just with OGs and taxids
all_interactions_simple = [] #All interactions in the meta-interactome, but just with OGs
consensus_interactions = [] #Consensus interactome interactions first, then associated data. Get written to
output file.
all_annotations = [] #Annotations in file - a bit inefficient to load the whole thing but more searchable this way
consensus_annotations = {} #Dictionary to store functional category and descriptions of OG interactors.

#Load the meta-interactome file, removing true cross-species interactions
for line in metafile:
    one_interaction = ((line.rstrip()).split("\t"))
    taxid_A = (((one_interaction[9].split("|"))[0]).lstrip("taxid:")).split("(")[0]
    taxid_B = (((one_interaction[10].split("|"))[0]).lstrip("taxid:")).split("(")[0]
    taxid_mismatch = 0 #Assume that the two taxids are the same by default
    if taxid_A != taxid_B: #If the two taxids aren't identical, they may still be related or may truly be cross-
species.
        #Cross-species PPI get removed.
        taxid_mismatch = 1
        if (taxid_species[taxid_A])[1] == (taxid_species[taxid_B])[1]: #Check if taxids share a parent
            taxid_mismatch = 0
        if (taxid_species[taxid_A])[1] == taxid_B or (taxid_species[taxid_B])[1] == taxid_A: #Check for
parent-child relationship
            taxid_mismatch = 0
    if taxid_mismatch != 1:
        all_interactions_taxids.append([one_interaction[42], one_interaction[43], taxid_A, taxid_B])
        all_interactions_simple.append([one_interaction[42], one_interaction[43]])

#First pass: create a list of unique interactors and interactions, using OG IDs
print("Finding unique interactors and interactions. Interactions found:")
cons_interaction_count = 0

for interaction in all_interactions_simple:
    interaction_rev = [interaction[1], interaction[0]]
    new_interaction = 0
    for interactor in interaction: #Interactor A or B's OG or ID if no OG mapped
        if interactor not in consensus_interactors:
            consensus_interactors.append(interactor)
    if interaction not in consensus_interactions and interaction_rev not in consensus_interactions:
        new_interaction = 1
    if new_interaction == 1:

```

```

        cons_interaction_count = cons_interaction_count + 1
    if cons_interaction_count % 100 == 0:
        sys.stdout.flush()
        sys.stdout.write(".")
    if cons_interaction_count % 1000 == 0:
        sys.stdout.flush()
        sys.stdout.write(str(cons_interaction_count))
    consensus_interactions.append(interaction)

#Second pass: count the number of interactions contributing to each consensus
#Compare taxids across interactions to see how many different sources interaction is seen for (i.e., X different
species)
#Add counts to each item in consensus_interactions
#The first count is the total occurrence of the given interaction across the full meta-interactome
#The second count is the number of different, unique taxids (species or at least distant strains) corresponding to
the interaction

print("\nCounting interaction contributions. This may take a while.")
print("Consensus interactions checked, out of %s:" % (len(consensus_interactions)))

all_consensus_taxids = []
con_interactions_counted = 0
for interaction in consensus_interactions:
    con_interactions_counted = con_interactions_counted + 1
    if con_interactions_counted % 100 == 0:
        sys.stdout.flush()
        sys.stdout.write(".")
    if con_interactions_counted % 1000 == 0:
        sys.stdout.flush()
        sys.stdout.write(str(con_interactions_counted))
    interaction_sources = []          #The list of taxids found to correspond to this interaction.
    original_count = 0
    #This gets a bit complicated.
    for original_interaction in all_interactions_taxids:      #For each interaction in the set of all (not OG-
compressed consensus) meta-interactome interactions...
        original_interaction_slim = original_interaction[0:2]
        original_interaction_slim_rev = [original_interaction[1], original_interaction[0]]
        if interaction == original_interaction_slim or interaction == original_interaction_slim_rev:      #If
the original interaction OR its reverse matches the consensus interaction...

```



```

original_count = original_count +1      #Add to the count of this interaction across the
meta-interactome.
for taxid in original_interaction[2:4]: #For both taxids corresponding to the meta-
interactome interaction...
    if taxid not in interaction_sources and taxid != "-": #If the taxid isn't in the
source taxids for this interaction yet...and isn't empty...
        if (taxid_species[taxid])[1] not in interaction_sources:      #Check to see
if the sources contain the taxid's parent taxid (if so, it's redundant)
            for source in interaction_sources:
                if (taxid_species[source])[1] == taxid: #Check to see if
taxid is a parent of existing sources (if so, remove children and just use parent)
                    interaction_sources.remove(source)
                if (taxid_species[source])[1] == (taxid_species[taxid])[1]:
#Check if taxids share a parent (if so, use parent taxid and drop children)
                    taxid = (taxid_species[source])[1]      #The problem
here is that the parent taxid may not be in taxid_species since it's new to us

                    #So we look it up and add it!
                    target_handle = Entrez.efetch(db="Taxonomy",

id=str(taxid), retmode="xml")

                    target_records = Entrez.read(target_handle)
                    taxid_name = target_records[0]["ScientificName"]
                    taxid_parent = target_records[0]["ParentTaxId"]
                    taxid_species[taxid] = [taxid_name, taxid_parent]
                    #This really should be its own function to limit

redundancy

                    interaction_sources.remove(source)
                    interaction_sources.append(taxid)
for source in interaction_sources:      #List all the taxids used across the consensus - does NOT care about
parent/child relationships
    if source not in all_consensus_taxids:
        all_consensus_taxids.append(source)
interaction.append(str(original_count))
interaction.append(str(len(interaction_sources)))
interaction.append(" ".join(interaction_sources))
#print(interaction)

#Third pass: get the functional categories and descriptions of all interactors

```

```

print("\nAdding interactor annotations.")
for input_ann_file in annotation_file_list:
    try:
        ann_file = open(input_ann_file)
    except IOError as e:
        print("I/O error({0}): {1}".format(e.errno, e.strerror))
    for line in ann_file:
        all_annotations.append((line.rstrip()).split("\t"))
    ann_file.close()

multiple_og_count = 0    #The count of interactors mapping to >1 OG. Are treated as single OGs as this may be
biologically meaningful
annotation_count = 0
for interactor in consensus_interactors:
    annotation_count = annotation_count + 1
    if annotation_count % 100 == 0:
        sys.stdout.write(".")
    if annotation_count % 1000 == 0:
        sys.stdout.write(str(annotation_count))
    consensus_annotations[interactor] = ["NA", "NA"] #Should only happen if OG not in description file (e.g. if
it's unmapped to an OG)
    if "," in interactor:    #Meaning it maps to multiple OGs, so we need to annotate all of them
        #print(interactor)
        this_mult_og_count = 0    #Keeps track of multiple OG sets. Usually just two or three different OGs at
most.

        consensus_annotations[interactor] = ["", ""]
        multiple_og_count = multiple_og_count + 1
        multiple_ogs = interactor.split(",")
        for og in multiple_ogs:
            this_mult_og_count = this_mult_og_count + 1
            for annotation in all_annotations:
                if og == annotation[1]:
                    #Concatenate each FuncCat, separated by |
                    (consensus_annotations[interactor])[0] = (consensus_annotations[interactor])
[0] + annotation[4]

                    if this_mult_og_count != len(multiple_ogs):
                        (consensus_annotations[interactor])[0] =
(consensus_annotations[interactor])[0] + "|"

                    #Concatenate each description, separated by |

```

```

                                (consensus_annotations[interactor])[1] = (consensus_annotations[interactor])
[1] + annotation[5]

                                if this_mult_og_count != len(multiple_ogs):
                                    (consensus_annotations[interactor])[1] =
(consensus_annotations[interactor])[1] + "|"
                                break

                                else:
                                    for annotation in all_annotations:
                                        if interactor == annotation[1]:
                                            consensus_annotations[interactor] = [annotation[4], annotation[5]] #FuncCat and
description
                                            break
                                for interaction in consensus_interactions:
                                    for interactor in interaction[0:2]:
                                        interaction.append("\t".join(consensus_annotations[interactor]))

print("\nConsensus meta-interactome involves " +
      "%s interactors and %s interactions." % (len(consensus_interactors), cons_interaction_count))
print("It involves %s unique taxids, " % (len(all_consensus_taxids)) +
      "though some may be closely related.")
print("%s interactors map to more than one OG." % multiple_og_count)

print("Writing consensus meta-interactome file.")
for interaction in consensus_interactions:
    consensus_network_file.write("\t".join(interaction) + "\n")
consensus_network_file.close()

cons_stats_filename = "cons_statistics_" + nowstring + ".txt"
cons_stats_file = open(cons_stats_filename, "w")
stats_header = ("Unique interactors\tInteractions\tTaxids\n")
cons_stats_file.write(stats_header)
cons_statistics = []
for cons_stat in [len(consensus_interactors), cons_interaction_count, len(all_consensus_taxids)]:
    cons_statistics.append(str(cons_stat))
cons_stats_file.write("\t".join(cons_statistics))
print("Wrote consensus meta-interactome statistics to " + cons_stats_filename)
cons_stats_file.close()

return consensus_network_filename

```

```

def merge_data(list_of_filenames):

    nowstring = (date.today()).isoformat()
    merged_file_name = "interactions" + nowstring + ".txt"
    merged_file = open(merged_file_name, "w")

    for item in list_of_filenames:
        this_file = open(item)
        line_count = 0
        for line in this_file:
            write_ok = 1
            line_count = line_count + 1
            line_contents = ((line.rstrip()).split("\t"))
            for interactor in line_contents[0:2]:
                if interactor == "-":
                    print("Empty interactor in %s in line %s" + str() % (item, line_count))
                    write_ok = 0
                    #Unmapped interactors might be denoted with a -. Don't add them.
            if len(line_contents) != 42:
                print("Format problem in %s line %s" % (item, line_count))
                write_ok = 0
                #Just checking to see if the right number of columns are there
                #Won't write problem lines to the merged file
            if write_ok == 1:
                merged_file.write(line)
        this_file.close()
    return merged_file_name

def subgraph_expansion(metafile, consensusfile):
    print("\nSubset expansion will filter consensus interactions by functional category" +
          " and by conservation across taxonomic groups.\n" +
          "It will produce a set of subgraphs, where each graph involves a consensus" +
          " interactor and ALL of its interactions in the meta-interactome.\n" +
          "These graphs will contain taxonomy annotations for each interaction" +
          " and can be split in network analysis software, e.g. Cytoscape.\n")
    print("Functional categories:\n"
          "INFORMATION STORAGE AND PROCESSING\n"
          "[J] Translation, ribosomal structure and biogenesis\n")

```

```

"[A] RNA processing and modification\n"
"[K] Transcription\n"
"[L] Replication, recombination and repair\n"
"[B] Chromatin structure and dynamics\n"
"CELLULAR PROCESSES AND SIGNALING\n"
"[D] Cell cycle control, cell division, chromosome partitioning\n"
"[Y] Nuclear structure\n"
"[V] Defense mechanisms\n"
"[T] Signal transduction mechanisms\n"
"[M] Cell wall/membrane/envelope biogenesis\n"
"[N] Cell motility\n"
"[Z] Cytoskeleton\n"
"[W] Extracellular structures\n"
"[U] Intracellular trafficking, secretion, and vesicular transport\n"
"[O] Posttranslational modification, protein turnover, chaperones\n"
"METABOLISM\n"
"[C] Energy production and conversion\n"
"[G] Carbohydrate transport and metabolism\n"
"[E] Amino acid transport and metabolism\n"
"[F] Nucleotide transport and metabolism\n"
"[H] Coenzyme transport and metabolism\n"
"[I] Lipid transport and metabolism\n"
"[P] Inorganic ion transport and metabolism\n"
"[Q] Secondary metabolites biosynthesis, transport and catabolism\n"
"POORLY CHARACTERIZED\n"
"[R] General function prediction only\n"
"[S] Function unknown\n")

```

```

func_filter = raw_input("Filter interactors for which functional category? (Type X for interactors of unknown function.)\n")

```

```

search_unknowns = 0

```

```

if func_filter in ["x", "X"]:

```

```

    search_unknowns = 1

```

```

    print("Filtering for interactors of unknown function. Interactors marked NA will not be included.")

```

```

consensus_interactions = []      #Contains whole line (one interaction) from consensus

```

```

consensus_interactions_filtered = [] #Interactions filtered by FuncCat

```

```

consensus_interactions_taxfilt = []      #Interactions filtered by FuncCat and taxids

```

```

consensus_interactors_filtered = {}      #Contains interactor, FuncCat, and description (filtered by FuncCat)

```

```

consensus_interactors_taxfilt = {}      #Contains interactor, FuncCat, and description (filtered by FuncCat and
                                         #by participation in an interaction passing
the taxid filter)
expanded_interactions = {}              #Keys are consensus interactors. Values are all unique proteins (and
sources) they interact with.

                                         #Actually a dict of lists of lists. Fun.
max_taxon_range = 1                    #The greatest count of different taxids per interaction, across the whole consensus
all_interactions = []                  #All interactions in the meta-interactome
protein_annotations = {}                #Annotations (from IntAct) for unique proteins. No FuncCats here.

print("Filtering consensus interactors by function.")
consensusfile.seek(0)                  #In case we've been using the file already
for line in consensusfile:              #Filter interactors by FuncCat
    one_consensus_interaction = (line.rstrip()).split("\t")
    consensus_interactions.append(one_consensus_interaction)

    if one_consensus_interaction[5] != "NA":      #For interactor A
        if search_unknowns == 1:
            if "R" in one_consensus_interaction[5] or "S" in one_consensus_interaction[5]:
                consensus_interactors_filtered[one_consensus_interaction[0]] =
[one_consensus_interaction[5], one_consensus_interaction[6]]
            else:
                if func_filter in one_consensus_interaction[5]:
                    consensus_interactors_filtered[one_consensus_interaction[0]] =
[one_consensus_interaction[5], one_consensus_interaction[6]]
                if one_consensus_interaction[7] != "NA":      #For interactor B
                    if search_unknowns == 1:
                        if "R" in one_consensus_interaction[7] or "S" in one_consensus_interaction[7]:
                            consensus_interactors_filtered[one_consensus_interaction[1]] =
[one_consensus_interaction[7], one_consensus_interaction[8]]
                        else:
                            if func_filter in one_consensus_interaction[7]:
                                consensus_interactors_filtered[one_consensus_interaction[1]] =
[one_consensus_interaction[7], one_consensus_interaction[8]]

consensusfile.close()                  #May not want to close file if we plan on doing multiple filters during same session.

for interaction in consensus_interactions:
    if interaction[0] in consensus_interactors_filtered or interaction[1] in consensus_interactors_filtered:

```

```

        consensus_interactions_filtered.append(interaction)
        if interaction[3] > max_taxon_range:
            max_taxon_range = interaction[3]

print("The maximum for this filter will be " + str(max_taxon_range) + " different taxids.")
tax_filter = raw_input("Select for at least how many different taxonomic groups?\n")

for interaction in consensus_interactions_filtered:
    if interaction[3] >= tax_filter:
        consensus_interactions_taxfilt.append(interaction)

        if interaction[5] != "NA":          #For interactor A
            if search_unknowns == 1:
                if "R" in interaction[5] or "S" in interaction[5]:
                    consensus_interactors_taxfilt[interaction[0]] = [interaction[5],
interaction[6]]
                else:
                    if func_filter in interaction[5]:
                        consensus_interactors_taxfilt[interaction[0]] = [interaction[5],
interaction[6]]
            if interaction[7] != "NA":          #For interactor B
                if search_unknowns == 1:
                    if "R" in interaction[7] or "S" in interaction[7]:
                        consensus_interactors_taxfilt[interaction[1]] = [interaction[7],
interaction[8]]
                else:
                    if func_filter in interaction[7]:
                        consensus_interactors_taxfilt[interaction[1]] = [interaction[7],
interaction[8]]

print("Generated list of filtered consensus interactors. Searching meta-interactome.")

for line in metafile:    #Set up the meta-interactome file first
    one_interaction = (line.rstrip()).split("\t")
    all_interactions.append(one_interaction)    #This is just the raw interaction list at this point

metafile.close()

for interactor in consensus_interactors_taxfilt:

```

```

        expanded_interactions[interactor] = []
        for interaction in all_interactions:      #Search meta-interactome for matching interactions; return unique all
proteins and corresponding organisms
            if interaction[42] == interactor:
                taxid = (((((interaction[10].split("|"))[0]).lstrip("taxid:")).split("(")[0])
                protein = interaction[1].lstrip("uniprotkb:")
                if protein not in protein_annotations:
                    protein_annotations[protein] = [interaction[23], interaction[43]]
                protein_and_source = [protein, taxid]
                (expanded_interactions[interactor]).append(protein_and_source)
            if interaction[43] == interactor:
                if interaction[0] != interaction[1]:      #Avoid adding self-interactions twice.
                    taxid = (((((interaction[9].split("|"))[0]).lstrip("taxid:")).split("(")[0])
                    protein = interaction[0].lstrip("uniprotkb:")
                    if protein not in protein_annotations:
                        protein_annotations[protein] = [interaction[22], interaction[42]]
                    protein_and_source = [protein, taxid]
                    (expanded_interactions[interactor]).append(protein_and_source)

nowstring = (date.today()).isoformat()
subgraph_file_name = "subgraph_expansion_" + func_filter + "_" + nowstring + ".txt"
subgraph_node_file_name = "subgraph_expansion_" + func_filter + "_nodes_" + nowstring + ".txt"
subgraph_file = open(subgraph_file_name, "w")
subgraph_node_file = open(subgraph_node_file_name, "w")

#print(consensus_interactors_taxfilt)
#print(expanded_interactions)

print("Writing subgraph expansion file and node annotation file.")
for consensus_interactor in expanded_interactions:
    for interaction in expanded_interactions[consensus_interactor]:
        #print(consensus_interactor + "\t" + "\t".join(interaction))
        subgraph_file.write(consensus_interactor + "\t" + "\t".join(interaction) + "\n")

    #Protein annotations are kind of a mess but that's because the interaction data table combines interactor annotations
into single columns.
    #It's also difficult to know what kind of annotation to expect.
    #All are included here, for now.

```



```

    for interactor in consensus_interactors_taxfilt:
        subgraph_node_file.write(interactor + "\t" + "\t".join(consensus_interactors_taxfilt[interactor]) + "\t-\n")
    for protein in protein_annotations:
        subgraph_node_file.write(protein + "\t-\t" + "\t".join(protein_annotations[protein]) + "\n")

    print("Done.")

def predict_interactome(mapping_file_list, metafile, consensusfile):
    cwd = os.getcwd()
    storage_path = "proteomes"
    if not os.path.isdir(storage_path):
        try:
            os.mkdir(storage_path)
            print("Setting up proteome directory.")
        except OSError:
            if not os.path.isdir(storage_path):
                raise

    pred_interactome_path = "predicted_interactomes"
    if not os.path.isdir(pred_interactome_path):
        try:
            os.mkdir(pred_interactome_path)
            print("Setting up predicted interactome directory.")
        except OSError:
            if not os.path.isdir(pred_interactome_path):
                raise

    getting_proteomes = 1
    #Can retrieve proteome entries from Uniprot and will map to OGs.
    while getting_proteomes == 1:
        get_new_proteomes = raw_input("Get a proteome from Uniprot? (Y/N)\n")
        if get_new_proteomes in ["Y", "y"]:
            get_a_proteome()
            #run get_a_proteome() method
        else:
            print("Will now map proteomes to OGs.")
            break

    os.chdir(storage_path)
    proteome_list = glob.glob('proteome_raw_*.txt') #Raw proteomes, from Uniprot, in list format, labeled with taxid
    os.chdir("../")

```

```

map_dict = {}    #Dictionary for Uniprot to OG maps

if len(proteome_list) > 0:    #Only need the OG map if we have raw proteomes to be processed
    print("Setting up protein to OG maps.")
    for input_map_file in mapping_file_list:
        try:
            map_file = open(input_map_file)
        except IOError as e:
            print("I/O error({0}): {1}".format(e.errno, e.strerror))
        for line in map_file:
            one_map = ((line.rstrip()).split("\t"))
            map_dict[one_map[0]] = one_map[1]
        map_file.close()

for proteome_filename in proteome_list:    #Map all available raw proteomes to OGs.
    #Proteins without OG mappings retain their Uniprot IDs but we keep track of it in an extra column, too
    os.chdir(storage_path)
    print("Mapping proteins in " + proteome_filename)
    proteome_proteins = []
    proteome_map_filename = proteome_filename.replace("raw", "map")
    try:
        proteome_file = open(proteome_filename)
        proteome_map_file = open(proteome_map_filename, "w")
    except IOError as e:
        print("I/O error({0}): {1}".format(e.errno, e.strerror))
    for line in proteome_file:
        one_protein = line.rstrip()
        proteome_proteins.append(one_protein)
    total_proteins = 0
    total_proteins_mapped = 0
    for protein in proteome_proteins:
        og_mapped = 0    #All proteins are unmapped to OGs by default
        total_proteins = total_proteins + 1
        if protein in map_dict:
            og_mapped = 1
            total_proteins_mapped = total_proteins_mapped + 1
            proteome_map_file.write(map_dict[protein] + "\t" + protein + "\t" + str(og_mapped) + "\n")
        else:

```

```

        proteome_map_file.write(protein + "\t" + protein + "\t" + str(og_mapped) + "\n")

    proteome_file.close()
    os.remove(proteome_filename)    #Remove the raw file as it's redundant now.

    print(proteome_filename + " contains " + str(total_proteins) + " proteins. "
          + str(total_proteins_mapped) + " map to OGs.")
    if total_proteins_mapped == 0:
        print("WARNING: No proteins in this proteome map to OGs.")
    os.chdir("..")

os.chdir(storage_path)
proteome_map_list = glob.glob('proteome_map_*.txt')    #Proteomes mapped to eggNOG OGs, labeled with taxid
os.chdir("..")

#Uses Entrez here for more info about taxid corresponding to proteome.
print("\nAvailable proteome maps:")
taxid_context = {}    #We'll keep the taxonomy information for later.
for proteome_filename in proteome_map_list:
    taxid = (proteome_filename.split("_"))[2].rstrip(".txt")
    target_handle = Entrez.efetch(db="Taxonomy", id=str(taxid), retmode="xml")
    target_records = Entrez.read(target_handle)
    #print(target_records)
    taxid_name = target_records[0]["ScientificName"]
    taxid_parent = target_records[0]["ParentTaxId"]
    taxid_division = target_records[0]["Division"]
    if taxid_division != "Bacteria" and useViruses == False:
        print(taxid_name + "\t\t" + proteome_filename + "\tNOTE: Not Bacteria! May not work well with
bacterial consensus networks.")
    if taxid_division == "Viruses" and useViruses == True:
        print(taxid_name + "\t\t" + proteome_filename + "\tNOTE: This is a viral proteome. Ensure your meta-
interactome uses viral proteins.")
    else:
        print(taxid_name + "\t\t" + proteome_filename)
    taxid_context[taxid] = [taxid_name, taxid_parent, taxid_division]    #This is critical as we'll need it
shortly

#Also retrieve taxid details from the taxid context file.
taxid_context_filenames = glob.glob("taxid_context*.txt")

```

```

if len(taxid_context_filenames) > 1:
    print("More than one taxid context file found. Check for duplicates.")
    return None
if len(taxid_context_filenames) == 0:
    print("Cannot find taxid context file. Rebuild meta-interactome.")
    return None
taxid_context_file = open(taxid_context_filenames[0])
for line in taxid_context_file:
    one_context = (line.rstrip()).split("\t")
    this_taxid = one_context[0]
    taxid_name = one_context[1]
    taxid_parent = one_context[2]
    taxid_division = one_context[3]
    taxid_context[this_taxid] = [taxid_name, taxid_parent, taxid_division]

#Now that we have proteomes, we can use them to predict interactomes.

continue_prediction = raw_input("Predict interactomes for all above? (Y/N)\n")
if continue_prediction in ["Y", "y"]:
    print("Predicting interactomes for the above species.")
else:
    return None

print("Loading meta-interactome files.")          #Only uses the consensus right now, but full meta-interactome may be
needed
consensus_interactions = []                      #if we want to filter by spoke expansion or know
individual proteins
all_interactions = []

for line in consensusfile:
    one_consensus_interaction = (line.rstrip()).split("\t")
    consensus_interactions.append(one_consensus_interaction)
consensusfile.close()

for line in metafile:
    one_interaction = (line.rstrip()).split("\t")
    all_interactions.append(one_interaction)      #This is just the raw interaction list at this point
metafile.close()

```

```

#Interactome prediction starts here, iterating through each OG-mapped proteome.
interactome_stats = {} #Uses taxid as key

for proteome_filename in proteome_map_list: #Go through each of the available OG-mapped proteomes
    print("\nPredicting interactome for " + proteome_filename + ".")
    taxid = ((proteome_filename.split("_"))[2]).rstrip(".txt")
    this_proteome_map = {} #A dictionary of OGs to multiple proteins, since >1 protein may map to an OG.
    this_proteome = [] #A list of just proteins
    this_og_eome = [] #A list of just OGs in the proteome
    this_pred_interactome = [] #Actually the interactome at any one time - the whole prediction is written
to file
    this_pred_interactome_detailed = [] #The same interactome, but with contextual details
    #It will also include a prediction category.
    os.chdir(storage_path)
    proteome_map_file = open(proteome_filename)
    for line in proteome_map_file:
        contents = (line.rstrip()).split("\t")
        one_protein = contents[1]
        one_og = contents[0]
        if one_og not in this_proteome_map:
            this_proteome_map[one_og] = [one_protein]
        else:
            this_proteome_map[one_og].append(one_protein)
        this_proteome.append(one_protein) #Each line in the input should already contain a unique
protein ID
        if one_og not in this_og_eome:
            this_og_eome.append(one_og) #Should be the same as the keys in this_proteome_map
    proteome_map_file.close()

    os.chdir("../")
    os.chdir(pred_interactome_path)
    pred_interactome_filename = proteome_filename.replace("proteome_map", "pred_interactome")
    pred_interactome_file = open(pred_interactome_filename, "w")

    pred_ppi_count = 0 #The count of PPI from predictions
    exp_ppi_count = 0 #The count of PPI from experimental results, counting spoke expansion

    #First pass: check the meta-interactome for exact match PPI
    #This is mostly to account for proteins without OG matches, but we count them all

```

```

#as we want to distinguish between interactions seen already and new predictions.
#Like with building the consensus set, we need to check for related taxids.

print("Checking for experimental interactions.")
for interaction in all_interactions:
    same_species = 0          #Well, not the same, but same as the target species OR related
    parent_taxid = taxid_context[taxid][1]
    taxid_A = (((((interaction[9].split("|"))[0]).lstrip("taxid:")).split("(")[0])
    taxid_B = (((((interaction[10].split("|"))[0]).lstrip("taxid:")).split("(")[0])
    if taxid == taxid_A or parent_taxid == taxid_A: #If taxids are the same as target or its parent
        if taxid == taxid_B or parent_taxid == taxid_B:
            same_species = 1
    elif taxid == taxid_A or taxid == taxid_context[taxid_A][1]: #If taxids are child of target
        if taxid == taxid_B or taxid == taxid_context[taxid_B][1]:
            same_species = 1
    elif taxid == taxid_A or parent_taxid == taxid_context[taxid_A][1]: #If taxids share parent
        if taxid == taxid_B or parent_taxid == taxid_context[taxid_B][1]:
            same_species = 1
    #May throw KeyError here, indicating taxid not in taxid_context - look up if needed?
    if same_species == 1:
        proteinA = interaction[0].lstrip("uniprotkb:")
        proteinB = interaction[1].lstrip("uniprotkb:")
        ogA = interaction[42]
        ogB = interaction[43]
        unique_interaction = [proteinA, proteinB, ogA, ogB]
        unique_interaction_detailed = [proteinA, proteinB, ogA, ogB, "Experimental"]
        if unique_interaction not in this_pred_interactome:
            exp_ppi_count = exp_ppi_count + 1
            if exp_ppi_count % 10 == 0:
                sys.stdout.write(".")
            if exp_ppi_count % 100 == 0:
                sys.stdout.write(str(exp_ppi_count))
            this_pred_interactome.append(unique_interaction)
            this_pred_interactome_detailed.append(unique_interaction_detailed)
sys.stdout.write(str(exp_ppi_count))

print("\nMaking interaction predictions.")
#Second pass: make predictions based on OGs and the consensus interactome.
#That is, if two proteins interact, predict all proteins in their two OGs interact.

```

```

#All experimental interactions should be covered in the consensus, so don't care about species here
#Don't need to handle protein vs. protein as we should have seen it in the meta-interactome already

for interaction in consensus_interactions:
    if interaction[0] in this_og_eome and interaction[1] in this_og_eome:    #Check for OG vs. OG first
        for proteinA in this_proteome_map[interaction[0]]:                #Expand interaction to all possible
proteins with OG matches
            for proteinB in this_proteome_map[interaction[1]]:
                unique_interaction = [proteinA, proteinB, interaction[0], interaction[1]]
                unique_interaction_detailed = [proteinA, proteinB, interaction[0],
interaction[1], "Predicted"]

                if unique_interaction not in this_pred_interactome:
                    pred_ppi_count = pred_ppi_count + 1
                    this_pred_interactome.append(unique_interaction)
                    this_pred_interactome_detailed.append(unique_interaction_detailed)
                    if pred_ppi_count % 10 == 0:
                        sys.stdout.write(".")
                    if pred_ppi_count % 100 == 0:
                        sys.stdout.write(str(pred_ppi_count))
            elif interaction[0] in this_og_eome and interaction[1] in this_proteome:    #Check if it's an OG
and a protein
                for proteinA in this_proteome_map[interaction[0]]:        #Expand interaction to all possible
proteins with OG matches
                    proteinB = interaction[1]
                    unique_interaction = [proteinA, proteinB, interaction[0], interaction[1]]
                    unique_interaction_detailed = [proteinA, proteinB, interaction[0], interaction[1],
"Predicted"]

                    if unique_interaction not in this_pred_interactome:
                        pred_ppi_count = pred_ppi_count + 1
                        this_pred_interactome.append(unique_interaction)
                        this_pred_interactome_detailed.append(unique_interaction_detailed)
                        if pred_ppi_count % 10 == 0:
                            sys.stdout.write(".")
                        if pred_ppi_count % 100 == 0:
                            sys.stdout.write(str(pred_ppi_count))
            elif interaction[0] in this_proteome and interaction[1] in interaction[1] in this_og_eome:
#Check if it's a protein and an OG
                proteinA = interaction[0]

```

```

        for proteinB in this_proteome_map[interaction[1]]:      #Expand interaction to all possible
proteins with OG matches
            unique_interaction = [proteinA, proteinB, interaction[0], interaction[1]]
            unique_interaction_detailed = [proteinA, proteinB, interaction[0], interaction[1],
"Predicted"]

            if unique_interaction not in this_pred_interactome:
                pred_ppi_count = pred_ppi_count + 1
                this_pred_interactome.append(unique_interaction)
                this_pred_interactome_detailed.append(unique_interaction_detailed)
                if pred_ppi_count % 10 == 0:
                    sys.stdout.write(".")
                if pred_ppi_count % 100 == 0:
                    sys.stdout.write(str(pred_ppi_count))

sys.stdout.write(str(pred_ppi_count))

#Finally - get a few more protein and OG counts.
#These counts won't be right if we just use the proteome, as PPI may include related species
#Isn't a problem for predictions as they're all based off one proteome
#But for experimental results we just get every Uniprot ID
proteins_in_interactions = []
proteins_w_exp_ppi = []
proteins_w_pred_ppi = []
ogs_in_interactions = []
ogs_w_exp_int = []
ogs_w_pred_int = []
exp_og_int = []
pred_og_int = []
for interaction in this_pred_interactome_detailed:
    og_pair = interaction[2:4]
    rev_og_pair = [interaction[3], interaction[2]] #We don't care about interaction direction.
    if interaction[4] == "Experimental":
        for protein in interaction[0:2]:
            if protein not in proteins_w_exp_ppi:
                proteins_w_exp_ppi.append(protein)
        for og in og_pair:
            if og not in ogs_w_exp_int:
                ogs_w_exp_int.append(og)
    if og_pair not in exp_og_int and rev_og_pair not in exp_og_int:
        exp_og_int.append(og_pair)

```



```

for protein in this_proteome:
    both_interactors = 0
    if protein in interaction[0:2] and protein not in proteins_in_interactions:
        both_interactors = both_interactors + 1
        proteins_in_interactions.append(protein)
        if interaction[4] == "Predicted" and protein not in proteins_w_pred_ppi:
            proteins_w_pred_ppi.append(protein)
    if both_interactors == 2:
        break
for og in this_og_eome:
    both_interactors = 0
    if og in interaction[2:4] and og not in ogs_in_interactions:
        both_interactors = both_interactors + 1
        ogs_in_interactions.append(og)
        if interaction[4] == "Predicted" and og not in ogs_w_pred_int:
            ogs_w_pred_int.append(og)
    if both_interactors == 2:
        break
if og_pair not in exp_og_int and rev_og_pair not in pred_og_int:
    pred_og_int.append(og_pair)

proteins_not_in_interactions = len(this_proteome) - len(proteins_in_interactions)      #Just a count, here
ogs_not_in_interactions = len(this_og_eome) - len(ogs_in_interactions)
#interactome_stats contains statistics used in batch output. Contains:
#Name, taxid, Proteins, ProteinsNotInPPI, ProteinsWithExpPPI, ProteinsWithPredPPI,
#UniqueOGs, OGsWithoutInteractions, OGsWithExpInt, OGsWithPredInt, ExpOGIntNet, OGIntInPredNet
interactome_stats[taxid] = [taxid_context[taxid][0], taxid, str(len(this_proteome)),
str(proteins_not_in_interactions),
                                                                    str(len(proteins_w_exp_ppi)),
str(len(proteins_w_pred_ppi)), str(len(this_og_eome)),
                                                                    str(ogs_not_in_interactions),
str(len(ogs_w_exp_int)), str(len(ogs_w_pred_int)),
                                                                    str(len(exp_og_int)), str(len(pred_og_int))]

#This is just for testing.
'''
for interaction in all_interactions:
    taxid_A = (((((interaction[9].split("|"))[0]).lstrip("taxid:")).split("(")[0])
    taxid_B = (((((interaction[10].split("|"))[0]).lstrip("taxid:")).split("(")[0])

```

```

        if taxid == taxid_A or taxid == taxid_B:
            proteinA = interaction[0].lstrip("uniprotkb:")
            proteinB = interaction[1].lstrip("uniprotkb:")
            ogA = interaction[42]
            ogB = interaction[43]
            this_meta_interaction = [proteinA, proteinB, ogA, ogB]
            if this_meta_interaction not in this_pred_interactome:
                print(this_meta_interaction)
        ...

#Write interactome file.
for interaction in this_pred_interactome_detailed:
    pred_interactome_file.write("\t".join(interaction) + "\n")

print("\nFound " + str(exp_ppi_count) + " experimental interactions (including spoke" +
      " expansion) and made " + str(pred_ppi_count) + " interaction predictions" +
      " for " + taxid_context[taxid][0] + ".")

pred_interactome_file.close()
os.chdir("../")

#Once all the interactome predictions for all proteomes are done, do summary statistics.
nowstring = (date.today()).isoformat()
multi_inter_stats_file_name = "interactome_statistics_" + nowstring + ".txt"
multi_inter_stats_file = open(multi_inter_stats_file_name, "w")
os.chdir("predicted_interactomes")
interactome_filenames = glob.glob("pred_interactome_*.txt")

stats_outlines = []      #This is where the output for each species will go, one interactome per line
for filename in interactome_filenames:
    taxid = ((filename.rstrip(".txt")).split("_"))[2]
    outline = "\t".join(interactome_stats[taxid]) + "\n"
    stats_outlines.append(outline)

os.chdir("../")
#Text header
multi_inter_stats_file.write("Species\tTaxid\tProteins\tProteinsNotInPPI\tProteinsWithExpPPI\tProteinsWithPredPPI\t")

```

+

```

"UniqueOGs\tOGsWithoutInteractions\tOGsWithExpInt\tOGsWithPredInt\tOGIntInExpNet\tOGIntInPredNet\n")
    for outline in stats_outlines:
        multi_inter_stats_file.write(outline)

    print("\nWrote summary statistics for these interactomes to " + multi_inter_stats_file_name)
    print("\nComplete.\n")

def get_a_proteome():
    proteins_umbra.get_a_proteome()

def describe_consensus(consensusfile):
    cons_stats_filenames = glob.glob("cons_statistics_*.txt")
    if len(cons_stats_filenames) > 1:
        print("More than one consensus statistics file found. Check for duplicates.")
        return None
    if len(cons_stats_filenames) == 0:
        print("No consensus statistics file found. Will skip basic counts.")
    else:
        con_stats = open(cons_stats_filenames[0])
        for line in con_stats:
            print(line)
        print("\n")

    consensus_interactions = []
    for line in consensusfile:
        one_interaction = (line.rstrip()).split("\t")
        consensus_interactions.append(one_interaction)

    taxids_and_context = {}
    taxid_ref_list = glob.glob('taxid_context*.txt')
    if len(taxid_ref_list) > 1:
        sys.exit("Something went wrong - more than one taxid context file found.")
    if len(taxid_ref_list) == 0:
        sys.exit("Something went wrong - no taxid context file found.")
    taxid_ref_file = open(taxid_ref_list[0])
    for line in taxid_ref_file:
        content = ((line.rstrip()).split("\t"))
        taxids_and_context[content[0]] = [content[1], content[2], content[3]]

```

```

taxid_ref_file.close()

print("Top taxid contributions, in number of consensus interactions corresponding to the taxid.")
print("Name\tTaxid\tNumber of interactions")
all_taxids = {} #All taxids AND their counts.

for interaction in consensus_interactions:      #Check each interaction for contributing taxids
    these_sources = interaction[4].split()
    for taxid in these_sources:
        if taxid not in all_taxids:
            all_taxids[taxid] = 1
        all_taxids[taxid] = all_taxids[taxid] + 1

sorted_taxids = sorted(all_taxids.items(), key=operator.itemgetter(1), reverse=True)
top_taxids = sorted_taxids[0:15]

for taxid in top_taxids:
    taxid_only = taxid[0]
    taxid_name = taxids_and_context[taxid_only][0]
    print(taxid_name + "\t" + taxid_only + "\t" + str(taxid[1]))

def main():
    #Check for eggNOG mapping file and get if needed
    #Requires downloading several files and building new mapping file from them
    mapping_file_list = glob.glob('uniprot_og_maps*.txt')
    if len(mapping_file_list) > 2:
        sys.exit("Found more than one mapping file. Check for duplicates.")
    if len(mapping_file_list) == 0:
        print("No eggNOG mapping files found or they're incomplete. Rebuilding them.")
        get_eggnog_maps()
        mapping_file_list = glob.glob('uniprot_og_maps*.txt')

    #Check for eggNOG annotation file and get if needed
    annotation_file_list = glob.glob('*annotations.tsv')
    expected_filecount = 2
    if useViruses == True:
        expected_filecount = 3
    if len(annotation_file_list) > expected_filecount:
        sys.exit("Found more eggNOG annotation files than expected. Check for duplicates.")

```

```

if len(annotation_file_list) < expected_filecount:
    print("No eggNOG annotation files found or they're incomplete. Retrieving them.")
    get_eggnog_annotations()
    annotation_file_list = glob.glob('*annotations.tsv')

#Prompt for choice of protein interactions.
#May provide manually or may download, but downloaded set may not be filtered properly.
#Don't need to get interactions if we already have a meta-interactome.
meta_file_list = glob.glob('*metainteractome*.txt')
ppi_data_filename = ""
if len(meta_file_list) > 1:
    sys.exit("More than one meta-interactome found. Please use just one at a time.")
if len(meta_file_list) == 0:
    print("\nNo meta-interactome found.")
    while ppi_data_filename == "":
        ppi_data_option = raw_input("Retreive IntAct bacterial PPI or use local file(s) to build meta-
interactome?\n")

        "Enter:\n R for retrieval\n L for local file, \n M for multiple inputs, \n or X to quit.\n")
        if ppi_data_option in ["R", "r"]:
            #Downloads PPI data from IntAct server.
            #May not include all PPI available through HTTP IntAct interface.
            ppi_data_filename = "protein-interactions.tab"
            interaction_file_list = glob.glob(ppi_data_filename)
            if len(interaction_file_list) > 1:
                sys.exit("One protein interaction file at a time, please! Check for duplicates.")
            if len(interaction_file_list) == 0:
                print("No protein interaction file found. Retrieving it.")
                get_interactions()
                interaction_file_list = glob.glob(ppi_data_filename)

            try:
                interactionfile = open(interaction_file_list[0])
            except IOError as e:
                print("I/O error({0}): {1}".format(e.errno, e.strerror))

        if ppi_data_option in ["L", "l"]:
            #Uses a local file, usually a downloaded IntAct PPI set, in
PSI-MI Tab27 format

            print("Will use single local file. Note that it should be in PSI-MI TAB 2.7 format and have
no header row.")

            ppi_data_filename = raw_input("Please provide local filename.\n")
            interaction_file_list = glob.glob(ppi_data_filename)
            if len(interaction_file_list) == 0:

```

```

        sys.exit("Can't find a file with that filename.")
    if ppi_data_option in ["M", "m"]:          #Uses multiple local files in PSI-MI Tab27 format
        print("Will append multiple local files. Note that each should be in PSI-MI TAB 2.7 format
and have no header row.")

        adding_files = 1
        interaction_file_list = []
        while adding_files:
            ppi_data_filename = raw_input("Please provide local filename.\n")
            files_present = glob.glob(ppi_data_filename)
            if len(files_present) >0:
                interaction_file_list.append(ppi_data_filename) #Can be expanded easily later

            print("Added " + ppi_data_filename + " to input list.")
        else:
            print("Can't find a file with that filename. Didn't add.")
            ask_again = raw_input("Add another? Y/N\n")
            if ask_again in ["N", "n"]:
                adding_files = 0

        print("Using the following inputs for the meta-interactome:\n")
        if len(interaction_file_list) == 0:
            sys.exit("Input list is empty. Exiting...")
        for item in interaction_file_list:
            print(item)
        print("Merging into a single file and checking for malformed interaction entries.")
        ppi_data_filename = merge_data(interaction_file_list)
    if ppi_data_option in ["X", "x"]:
        sys.exit("Exiting...")

#Load meta-interactome network file
#Needs to be built first.
new_meta = 0
if len(meta_file_list) == 0:
    build_meta_network = raw_input("Build a new meta-interactome? Y/N ")
    if build_meta_network in ["Y", "y"]:
        new_meta_result = build_meta(mapping_file_list, ppi_data_filename)
        new_meta_filename = new_meta_result[0]
        taxids_and_context = new_meta_result[1]
        new_meta = 1
    else:

```

```

        sys.exit("Meta-network needed. Exiting.")
try:
    if new_meta == 1:
        metafile = open(new_meta_filename)
    else:
        metafile = open(meta_file_list[0])
except IOError as e:
    print("I/O error({0}): {1}".format(e.errno, e.strerror))
print("\nUsing " + metafile.name + " as the meta-interactome network.")

#Load consensus network file
#Needs to be built first.
consensus_file_list = glob.glob('*consensus*.txt')
new_consensus = 0
if len(consensus_file_list) >1:
    sys.exit("One consensus network at a time, please!")
if len(consensus_file_list) == 0:
    print("No consensus network file found. Building one.")
    description_file = open("bactNOG.annotations.tsv")
    if new_meta == 1:
        #If we just build a meta-interactome we have taxid details already
        new_consensus_filename = build_consensus(metafile, annotation_file_list, taxids_and_context)
    else:
        #Otherwise we need to read taxid details from file - just rebuild dict from it
        taxid_ref_list = glob.glob('taxid_context*.txt')
        taxids_and_context = {}
        if len(taxid_ref_list) >1:
            sys.exit("Something went wrong - more than one taxid context file found.")
        if len(taxid_ref_list) == 0:
            sys.exit("Something went wrong - no taxid context file found.")
        taxid_ref_file = open(taxid_ref_list[0])
        for line in taxid_ref_file:
            content = ((line.rstrip()).split("\t"))
            taxids_and_context[content[0]] = [content[1], content[2], content[3]]
        taxid_ref_file.close()
        new_consensus_filename = build_consensus(metafile, annotation_file_list, taxids_and_context)
    new_consensus = 1
try:
    if new_consensus == 1:
        consensusfile = open(new_consensus_filename)
    else:

```

```

        consensusfile = open(consensus_file_list[0])
except IOError as e:
    print("I/O error({0}): {1}".format(e.errno, e.strerror))
print("\nUsing " + consensusfile.name + " as the consensus network.")

#Quit now or ask for next step.
requested = 0
while requested == 0:
    print("\n-----")
    request_next = raw_input("\nChoose from the following options.\n"
        "A: Generate expanded subgraphs of the consensus network, filtering by function.\n"
        "B: Generate a predicted interactome for one or more proteomes.\n"
        "C: Get statistics for the consensus meta-interactome.\n"
        "X: Exit.\n")
    if request_next in ["x", "X"]:
        sys.exit("Exiting...")
    if request_next in ["a", "A"]:
        subgraph_expansion(metafile, consensusfile)
    if request_next in ["b", "B"]:
        predict_interactome(mapping_file_list, metafile, consensusfile)
    if request_next in ["c", "C"]:
        describe_consensus(consensusfile)
    print("\nChoose from the list, please.")

if __name__ == "__main__":
    main()

sys.exit(0)

```

II.II.II *proteins_umbra.py*

```

#!/usr/bin/python
#proteins_umbra.py
'''

```


Downloads a reference proteome and assigns an orthologous group (OG) to each.
Uses eggNOG v.4.1 - or whatever the most recent version is.

REQUIRES: Biopython 1.65 or more recent

Needs ~5 GB of disk space for ID conversion files.

Needs an additional ~29 GB for mapping virus protein IDs.

(This is because it uses the full Uniprot ID mapping database,
which is excessive but more reliable than their mapping server
for large mapping requests)

INPUT: Downloads a reference proteome from Uniprot.

Produces OG maps if needed or uses those produced by Network_umbra.py.

OUTPUT:

'proteome_map_[taxid].txt' - on each line:

a single OG membership

the UniProtAC of the protein

a binary value indicating whether the protein maps to an OG (0 if no, 1 if yes)

...

```
import glob, gzip, operator, os, re, requests, sys, urllib2, zipfile
from Bio import Entrez
from bs4 import BeautifulSoup
from collections import Counter
from datetime import date
```

```
Entrez.email = 'caufieldjh@vcu.edu'
```

```
#Options
```

```
useViruses = True          #Option for using eggNOG's viral OGs. Requires the filters permitting only Bacteria to be modified
                             #Also requires the viral OGs to be downloaded and added.
                             #This option needs to be set True BEFORE the Uniprot to OG map is built or it won't
```

```
include proteins from viruses
```

```
#NOTE: Viruses are not currently in the eggNOG ID conversion file
```

```
#The eggNOG protein IDs vary from protein to protein but are often Uniprot IDs (mnemonic identifiers, i.e. A9J730_BPLUZ)
```

```
#We still check the ID conversion file for them in case it gets updated soon
```

```

useNonRefProteomes = True          #Option to search non-reference Uniprot proteomes
#Many of these proteomes have been made redundant in Uniprot
#and this script ignores redundant results, so they will not be seen

#Functions

def chunkit(input_seq, chunk_size):
    return (input_seq[position:position + chunk_size] for position in xrange(0, len(input_seq), chunk_size))

def get_eggnog_maps(version):
    #Download and unzip the eggNOG ID conversion file
    #Filters file to just Uniprot IDs; the resulting file is the map file.
    #One Uniprot ID may correspond to multiple OGs - e.g. COG1234,COG3810,COG9313.
    #these cases are considered OGs in their own right as this may indicate a pattern of conserved sequences on its own
    baseURL = "http://eggnogdb.embl.de/download/" + version + "/"
    convfilename = "eggnog4.protein_id_conversion.tsv.gz"    #File contains ALL database identifiers and corresponding
    proteins

    convfilepath = baseURL + convfilename
    outfilepath = convfilename[0:-3]
    dl_convfile = 1 #If 1, we need to download
    if os.path.isfile(convfilename): #Already have the compressed file, don't download
        print("Found compressed ID conversion file on disk: %s" % convfilename)
        decompress_convfile = 1
        dl_convfile = 0
    if os.path.isfile(outfilepath): #Already have the decompressed file, don't download
        print("Found ID conversion file on disk: %s" % outfilepath)
        decompress_convfile = 0
        dl_convfile = 0

    if dl_convfile == 1:
        print("Downloading ID mapping file - this file is large so this may take some time.")
        print("Downloading from %s" % convfilepath)
        response = urllib2.urlopen(convfilepath)
        filesize = response.info()['Content-Length']
        compressed_file = open(os.path.basename(convfilename), "w+b") #Start local compressed file
        chunk = 2097152
        totaldata = 0
        while 1:

```

```

        data = (response.read(chunk)) #Read two Mb at a time
        compressed_file.write(data)
        totaldata = totaldata + chunk
        if not data:
            print("\n%s file download complete." % convfilename)
            compressed_file.close()
            break
        sys.stdout.flush()
        sys.stdout.write("\r%s out of %s bytes" % (totaldata, filesize))
    decompress_convfile = 1

if decompress_convfile == 1:
    print("Decompressing map file. Lines written, in millions:")
    with gzip.open(convfilename) as infile: #Open that compressed file, read and write to uncompressed file
        outfile = open(outfilepath, "w+b")
        linecount = 0
        for line in infile:
            outfile.write(line)
            linecount = linecount + 1
            if linecount % 1000000 == 0:
                sys.stdout.flush()
                sys.stdout.write("\r%s" % (linecount/1000000))

        infile.close()
    newconvfilename = outfilepath
    outfile.close()

#Download and decompress member NOG files (at least 2 of them)
nogURL = baseURL + "data/NOG/"
nogfilename = "NOG.members.tsv.gz"
bactnogURL = baseURL + "data/bactNOG/"
bactnogfilename = "bactNOG.members.tsv.gz"
all_nog_locations = [[nogURL, nogfilename], [bactnogURL, bactnogfilename]]

if useViruses == True: #Need some additional files to handle viral proteins
    virnogURL = baseURL + "data/viruses/Viruses/"
    virnogfilename = "Viruses.members.tsv.gz"
    all_nog_locations.append([virnogURL, virnogfilename])
    up_baseURL = "ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/idmapping/"
    up_mapping_filename = "idmapping.dat.gz"

```

```

up_mapping_filepath = up_baseURL + up_mapping_filename
up_outfilepath = up_mapping_filename[0:-3]

dl_up_mapping_file = 1 #If 1, we need to download
if os.path.isfile(up_mapping_filename): #Already have the compressed file, don't download
    print("Found compressed Uniprot ID conversion file on disk: %s" % up_mapping_filename)
    decompress_up_mapping_file = 1
    dl_up_mapping_file = 0
if os.path.isfile(up_outfilepath): #Already have the decompressed file, don't download
    print("Found ID conversion file on disk: %s" % up_outfilepath)
    decompress_up_mapping_file = 0
    dl_up_mapping_file = 0

if dl_up_mapping_file == 1:
    print("\nDownloading Uniprot ID mapping file for viral protein mapping. Please wait as this file is
large.")

    print("Downloading from %s" % up_mapping_filepath)
    response = urllib2.urlopen(up_mapping_filepath)
    filesize = response.info()['Content-Length']
    compressed_file = open(os.path.basename(up_mapping_filename), "w+b") #Start local compressed file
    chunk = 2097152
    totaldata = 0
    while 1:
        data = (response.read(chunk)) #Read two Mb at a time
        compressed_file.write(data)
        if not data:
            print("\n%s file download complete." % up_mapping_filename)
            compressed_file.close()
            break
        sys.stdout.flush()
        sys.stdout.write("\r%s out of %s bytes" % (totaldata, filesize))
    decompress_up_mapping_file = 1

if decompress_up_mapping_file == 1:
    print("Decompressing Uniprot ID mapping file. Lines written, in millions:")
    with gzip.open(up_mapping_filename) as infile: #Open that compressed file, read and write to
uncompressed file
        outfile = open(up_outfilepath, "w+b")
        linecount = 0

```

```

        for line in infile:
            outfile.write(line)
            linecount = linecount + 1
            if linecount % 1000000 == 0:
                sys.stdout.flush()
                sys.stdout.write("\r%s" % (linecount/1000000))

        infile.close()
        outfile.close()

for location in all_nog_locations:
    baseURL = location[0]
    memberfilename = location[1]
    memberfilepath = baseURL + memberfilename
    outfilepath = memberfilename[0:-3]
    if os.path.isfile(memberfilename):
        print("\nFound compressed NOG membership file on disk: %s" % memberfilename)
        decompress_memberfile = 1
    if os.path.isfile(outfilepath):
        print("\nFound NOG membership file on disk: %s" % outfilepath)
        decompress_memberfile = 0
    else:
        print("\nDownloading NOG membership file - this may take some time.")
        print("Downloading from %s" % memberfilepath)
        response = urllib2.urlopen(memberfilepath)
        filesize = response.info()['Content-Length']
        compressed_file = open(os.path.basename(memberfilename), "w+b") #Start local compressed file
        chunk = 2097152
        totaldata = 0
        while 1:
            data = (response.read(chunk)) #Read two Mb at a time
            compressed_file.write(data)
            if not data:
                print("\n%s file download complete." % memberfilename)
                compressed_file.close()
                break
            sys.stdout.flush()
            sys.stdout.write("\r%s out of %s bytes" % (totaldata, filesize))
        decompress_memberfile = 1

```

```

if decompress_memberfile == 1:
    print("Decompressing NOG membership file %s" % memberfilename)
    #Done in chunks since it's a large file
    with gzip.open(memberfilename) as infile: #Open that compressed file, read and write to uncompressed
file
        outfile = open(outfilepath, "w+b")
        linecount = 0
        for line in infile:
            outfile.write(line)
            linecount = linecount + 1
            if linecount % 1000000 == 0:
                sys.stdout.flush()
                sys.stdout.write("\r%s" % (linecount/1000000))
        infile.close()
        outfile.close()

#Clean up by removing compressed files
print("\nRemoving compressed files.")
all_compressed_files = [convfilename, nogfilename, bactnogfilename]
if useViruses == True:
    for this_filename in [virnogfilename, up_mapping_filename]:
        all_compressed_files.append(this_filename)
for filename in all_compressed_files:
    if os.path.isfile(filename):
        os.remove(filename)

#Load and filter the ID conversion file as dictionary
print("Parsing ID conversion file. Lines read, in millions:")
with open(convfilename[0:-3]) as infile:
    id_dict = {} #Dictionary of eggNOG protein IDs as values and database IDs (UniprotAC) as keys
    #Gets filtered down to relevant database IDs (i.e., Uniprot IDs)
    linecount = 0
    for line in infile:
        linecount = linecount + 1
        line_raw = ((line.rstrip()).split("\t"))
        one_id_set = [line_raw[0] + "." + line_raw[1], line_raw[2], line_raw[3]] #Protein IDs are split for
some reason; merge them
        if "UniProt_AC" in one_id_set[2]:
            id_dict[one_id_set[1]] = one_id_set[0]

```

```

        if linecount % 1000000 == 0:
            sys.stdout.flush()
            sys.stdout.write("\r%s" % (linecount/1000000))
infile.close()

#Use filtered ID conversion input to map to NOG members
print("\nReading NOG membership files.")
all_nog_filenames = [nogfilename[0:-3], bactnogfilename[0:-3]]
if useViruses == True:
    all_nog_filenames.append(virnogfilename[0:-3])
nog_members = {}          #Dictionary of NOG ids with protein IDs as keys (need to split entries for each)
nog_count = 0
for filename in all_nog_filenames:
    temp_nog_members = {}  #We will have duplicates within each set but don't want to lose the information.
    print("Reading from %s" % filename)
    with open(filename) as infile:
        membercol = 5      #The column where the NOG members are
        if filename == virnogfilename[0:-3]:    #The virus members file has a different format as there is no
FuncCat column
            infile.readline()          #Skip the first line
            membercol = 4
            viral_ids = [] #A list of viral eggNOG protein IDs, some of which are Uniprot IDs to be
converted to ACs
        for line in infile:
            nog_count = nog_count + 1
            line_raw = ((line.rstrip()).split("\t"))
            nog_id = line_raw[1]
            line_members = line_raw[membercol].split(",")
            for protein_id in line_members:
                if filename == virnogfilename[0:-3]: #If Viruses, we need to convert IDs as they
aren't in the eggNOG ID conversion file.
                    if protein_id not in viral_ids:
                        viral_ids.append(protein_id)
                    if protein_id in temp_nog_members: #The same protein could be in more than one OG at
the same level
                        temp_nog_members[protein_id] = temp_nog_members[protein_id] + "," + nog_id
                    else:
                        temp_nog_members[protein_id] = nog_id
infile.close()

```

```

nog_members.update(temp_nog_members)

if useViruses == True:

    #We use three different dictionaries here.
    #The first is Uniprot IDs to UniprotACs (just for viral proteins)
    #The second is eggNOG IDs to Uniprot IDs.
    #The third is UniprotACs to eggNOG IDs - this is id_dict{} already.
    uniprotID_to_uniprotAC = {}
    eggnog_to_uniprotID = {}
    unmapped_ids = [] #eggNOG protein IDs which may not contain Uniprot IDs

    #We go through the viral protein IDs twice, first to get Uniprot IDs
    #and then to add them to id_dict.

    for viral_id in viral_ids:
        eggnog_to_uniprotID[viral_id] = (viral_id.split(".")[1]           #Remove the taxid
        #Some of the eggNOG IDs may not include UniprotIDs, but many do

    #This data file is too large to efficiently much of it in a dict.
    #Luckily we just got the IDs we need here to filter it
    print("Parsing Uniprot ID mapping file. Lines read, in millions:")
    with open(up_outfilepath) as infile:
        linecount = 0
        for line in infile:
            linecount = linecount +1
            line_raw = ((line.rstrip()).split("\t"))
            if line_raw[1] == "UniProtKB-ID" and line_raw[2] in eggnog_to_uniprotID.values():
                uniprotID_to_uniprotAC[line_raw[2]] = line_raw[0]
            if linecount % 1000000 == 0:
                sys.stdout.flush()
                sys.stdout.write("\r%s" % (linecount/1000000))
        infile.close()

    print("Finding identifiers for %s viral proteins." % len(viral_ids))

    for viral_id in viral_ids:
        upid = eggnog_to_uniprotID[viral_id]

```



```

        if upid in uniprotID_to_uniprotAC:
            upid_ac = uniprotID_to_uniprotAC[upid]
            id_dict[upid_ac] = viral_id
        else:
            unmapped_ids.append(viral_id)

    print("Done mapping viral proteins.")
    print("The following entries were not recognized as Uniprot IDs:")
    print(unmapped_ids)

#Get counts of how many identifiers we have now
upids_length = str(len(id_dict))
nogs_length = str(nog_count)
proteins_length = str(len(nog_members))

print("Mapping %s Uniprot IDs to %s NOGs through %s eggNOG protein IDs:" % (upids_length, nogs_length,
proteins_length))
upid_to_NOG = {}          #Conversion dictionary. Values are OGs, keys are UPIDs.
mapped_count = 0          #upids mapped to nog.
for upid in id_dict:
    if id_dict[upid] in nog_members:
        upid_to_NOG[upid] = nog_members[id_dict[upid]]
        mapped_count = mapped_count + 1
        if mapped_count % 100000 == 0:
            sys.stdout.flush()
            sys.stdout.write(".")
        if mapped_count % 1000000 == 0:
            sys.stdout.flush()
            sys.stdout.write(str(mapped_count/1000000))

#Use this mapping to build map file, named "uniprot_og_maps_*.txt"
print("\nWriting map file.")
nowstring = (date.today()).isoformat()
mapfilename = "uniprot_og_maps_" + nowstring + ".txt"
mapfile = open(mapfilename, "w+b")
for mapping in upid_to_NOG:
    mapfile.write(mapping + "\t" + upid_to_NOG[mapping] + "\n")    #Each line is a uniprot ID and an OG id
mapfile.close()

```

```

def get_eggnog_annotations():
    #Downloads and extracts the eggNOG NOG annotations.
    baseURLs = ["http://eggnogdb.embl.de/download/latest/data/bactNOG/",
"http://eggnogdb.embl.de/download/latest/data/NOG/"]
    bactannfilename = "bactNOG.annotations.tsv.gz" #The annotations for bacteria-specific NOGs
    lucaannfilename = "NOG.annotations.tsv.gz" #The annotations for other NOGs, but not bacteria-specific NOGs
    annfilenames = [bactannfilename, lucaannfilename]

    if useViruses == True:
        virannfilename = "Viruses.annotations.tsv.gz"
        baseURLs.append("http://eggnogdb.embl.de/download/latest/data/viruses/Viruses/")
        annfilenames.append(virannfilename)

    this_url = 0
    for annfilename in annfilenames:
        annfilepath = baseURLs[this_url] + annfilename
        this_url = this_url + 1
        outfilepath = annfilename[0:-3]
        if os.path.isfile(annfilename):
            print("Found compressed annotation file on disk: " + annfilename)
        else:
            response = urllib2.urlopen(annfilepath)
            filesize = response.info()['Content-Length']
            print("Downloading from " + annfilepath)
            compressed_file = open(os.path.basename(annfilename), "w+b") #Start local compressed file
            chunk = 2097152
            totaldata = 0
            while 1:
                data = (response.read(chunk)) #Read two Mb at a time
                compressed_file.write(data)
                if not data:
                    print("\n" + annfilename + " file download complete.")
                    compressed_file.close()
                    break
            sys.stdout.flush()
            sys.stdout.write("\r%s out of %s bytes" % (totaldata, filesize))

    print("Decompressing annotation file.")
    with gzip.open(annfilename) as infile: #Open that compressed file, read and write to uncompressed file

```

```

        file_content = infile.read()
        outfile = open(outfilepath, "w+b")
        outfile.write(file_content)
        infile.close()
    outfile.close()

print("\nRemoving compressed files.")
all_compressed_files = [bactannfilename, lucaannfilename]
if useViruses == True:
    all_compressed_files.append(virannfilename)
for filename in all_compressed_files:
    os.remove(filename)

def get_mapped_proteome(mapping_file_list):
    cwd = os.getcwd()
    storage_path = "proteomes"
    if not os.path.isdir(storage_path):
        try:
            os.mkdir(storage_path)
            print("Setting up proteome directory.")
        except OSError:
            if not os.path.isdir(storage_path):
                raise

    getting_proteomes = 1
    #Can retrieve proteome entries from Uniprot and will map to OGs.
    while getting_proteomes == 1:
        get_new_proteomes = raw_input("Get a proteome from Uniprot? (Y/N)\n")
        if get_new_proteomes in ["Y", "y"]:
            get_a_proteome()
            #run get_a_proteome() method
        else:
            print("Will now map proteomes to OGs.")
            break

    os.chdir(storage_path)
    proteome_list = glob.glob('proteome_raw_*.txt') #Raw proteomes, from Uniprot, in list format, labeled with taxid
    os.chdir("../")

    map_dict = {}
    #Dictionary for Uniprot to OG maps

```

```

if len(proteome_list) > 0:      #Only need the OG map if we have raw proteomes to be processed
    print("Setting up protein to OG maps.")
    for input_map_file in mapping_file_list:
        try:
            map_file = open(input_map_file)
        except IOError as e:
            print("I/O error({0}): {1}".format(e.errno, e.strerror))
        for line in map_file:
            one_map = ((line.rstrip()).split("\t"))
            map_dict[one_map[0]] = one_map[1]
        map_file.close()

unmapped_taxids = [] #These are the taxids without any OG mapping
for proteome_filename in proteome_list:      #Map all available raw proteomes to OGs.
    #Proteins without OG mappings retain their Uniprot IDs but we keep track of it in an extra column, too
    os.chdir(storage_path)
    print("Mapping proteins in " + proteome_filename)
    taxid = ((proteome_filename.split("_"))[2]).rstrip(".txt")
    proteome_proteins = []
    proteome_map_filename = proteome_filename.replace("raw", "map")
    try:
        proteome_file = open(proteome_filename)
        proteome_map_file = open(proteome_map_filename, "w")
    except IOError as e:
        print("I/O error({0}): {1}".format(e.errno, e.strerror))
    for line in proteome_file:
        one_protein = line.rstrip()
        proteome_proteins.append(one_protein)
    total_proteins = 0
    total_proteins_mapped = 0
    for protein in proteome_proteins:
        og_mapped = 0      #All proteins are unmapped to OGs by default
        total_proteins = total_proteins + 1
        if protein in map_dict:
            og_mapped = 1
            total_proteins_mapped = total_proteins_mapped + 1
            proteome_map_file.write(map_dict[protein] + "\t" + protein + "\t" + str(og_mapped) + "\n")
        else:
            proteome_map_file.write(protein + "\t" + protein + "\t" + str(og_mapped) + "\n")

```

```

proteome_file.close()
os.remove(proteome_filename)      #Remove the raw file as it's redundant now.

print(proteome_filename + " contains " + str(total_proteins) + " proteins. "
      + str(total_proteins_mapped) + " map to OGs.")
if total_proteins_mapped == 0:
    print("WARNING: No proteins in this proteome map to OGs.")
    unmapped_taxids.append(taxid)
os.chdir("..")

os.chdir(storage_path)
proteome_map_list = glob.glob('proteome_map_*.txt')      #Proteomes mapped to eggNOG OGs, labeled with taxid
os.chdir("..")

#Uses Entrez here for more info about taxid corresponding to proteome.
print("\nAvailable proteome maps:")
taxid_context = {}      #We'll keep the taxonomy information for later.
for proteome_filename in proteome_map_list:
    taxid = (proteome_filename.split("_"))[2].rstrip(".txt")
    target_handle = Entrez.efetch(db="Taxonomy", id=str(taxid), retmode="xml")
    target_records = Entrez.read(target_handle)
    #print(target_records)
    taxid_name = target_records[0]["ScientificName"]
    taxid_parent = target_records[0]["ParentTaxId"]
    taxid_division = target_records[0]["Division"]
    if taxid_division != "Bacteria" and useViruses == False:
        nameline = (taxid_name + "\t\t" + proteome_filename + "\tNOTE: Not Bacteria! May not work well with
bacterial consensus networks.")
    if taxid_division == "Viruses" and useViruses == True:
        nameline = (taxid_name + "\t\t" + proteome_filename + "\tNOTE: This is a viral proteome. Ensure your
meta-interactome uses viral proteins.")
    else:
        nameline = (taxid_name + "\t\t" + proteome_filename)
    if taxid in unmapped_taxids:
        nameline = "*** " + nameline
    print(nameline)
    taxid_context[taxid] = [taxid_name, taxid_parent, taxid_division]      #Not used at the moment

```

```

if len(unmapped_taxids) > 0:
    print("Maps marked with ** have no OG mappings.")

print("\nComplete.\n")

def get_a_proteome():    #Does what it says.    Much more organized than the rest of this since I wrote it a while ago.

    def get_search_url(query, fil):
        search_url = "http://www.uniprot.org/proteomes/?query=" + query + \
            "+redundant%3Ano&fil=" + fil + "&sort=score"

        return search_url

    def parse_search(up_input):
        search_results = []
        soup = BeautifulSoup(up_input, "lxml")
        for child in (soup.find_all('tr')):
            single_result = child.get_text("\t")
            search_results.append(single_result)
        del search_results[0:2]
        if len(search_results) == 0:
            print("No results found.")
            return None
        return search_results

    def get_proteome_url(entry, format_choice):
        proteome_url = "http://www.uniprot.org/uniprot/?sort=&desc=&query=proteome:" + entry + "&force=no&format=" +
format_choice
        return proteome_url

    def parse_proteome_entry(up_input):
        if not up_input:
            entry_text = "EMPTY"
        else:
            soup = BeautifulSoup(up_input, "lxml")
            entry_text = (soup.p.get_text())
        return entry_text

    def save_proteome(text,taxid):
        os.chdir("proteomes")

```

```

filename = "proteome_raw_" + str(taxid) + ".txt"
try:
    outfile = open(filename, 'wb')
except IOError as e:
    print("I/O error({0}): {1}".format(e.errno, e.strerror))
    sys.exit()
for line in text:
    outfile.write(line)
print("File written to " + filename)
outfile.close()
os.chdir("../")

#Retrieve proteomes on a query
query = (raw_input("Please specify a full or partial species name.\n")).rstrip()
ref_filter = "reference%3Ayes"
if useNonRefProteomes == True:
    ref_filter = ""
search_results_url = get_search_url(query, ref_filter) #Leave filter as "" to get non-reference proteomes too
                                                         #Other option: taxonomy%3ABacteria+%5B2%5D" for just
bacteria

search_response = requests.get(search_results_url)

#Output the query results
#print(search_response)
proteome_entries = parse_search(search_response.text)
if proteome_entries == None:
    return None
i = 0
print("Result\tAccession\tName")
for entry in proteome_entries:
    print(str(i) + "\t" + entry)
    i = i + 1

#Choose a single proteome and output to file
choice = raw_input('Please choose a search result.\n')
if not re.match("[0-9]*$", choice):
    print("Numbers only, please.")
    sys.exit()

```

```

chosen_entry = (proteome_entries[int(choice)]).split("\t")
print("Retrieving proteome for " + chosen_entry[1])
proteome_url = get_proteome_url(chosen_entry[0], "list") #Options include list, txt, tab
proteome_response = requests.get(proteome_url)
proteome_text = parse_proteome_entry(proteome_response.text)
if proteome_text == "EMPTY":
    print("Could not retrieve this proteome. It may be a redundant entry. See the Uniprot entry for %s." %
chosen_entry[0])
else:
    save_proteome(proteome_text, chosen_entry[2])

def main():
    #Check for eggNOG mapping file and get if needed
    #Requires downloading several files and building new mapping file from them
    mapping_file_list = glob.glob('uniprot_og_maps*.txt')
    if len(mapping_file_list) > 2:
        sys.exit("Found more than one mapping file. Check for duplicates.")
    if len(mapping_file_list) == 0:
        print("No eggNOG mapping files found or they're incomplete. Rebuilding them.")
        version = "latest"
        version = raw_input("Which eggNOG version would you prefer? Default is latest version.\n")
        if version not in ["4.5", "4.1", "4.0"]:
            version = "latest"
        else:
            version = "eggnog_" + version
        get_eggnog_maps(version)
        mapping_file_list = glob.glob('uniprot_og_maps*.txt')

    #Check for eggNOG annotation file and get if needed
    annotation_file_list = glob.glob('*annotations.tsv')
    expected_filecount = 2
    if useViruses == True:
        expected_filecount = 3
    if len(annotation_file_list) > expected_filecount:
        sys.exit("Found more eggNOG annotation files than expected. Check for duplicates.")
    if len(annotation_file_list) < expected_filecount:
        print("No eggNOG annotation files found or they're incomplete. Retrieving them.")
        get_eggnog_annotations()
        annotation_file_list = glob.glob('*annotations.tsv')

```



```

#Quit now or ask for next step.
requested = 0
while requested == 0:
    request_next = raw_input("\nChoose from the following options.\n"
        "A: Download a reference proteome and map to OGs.\n"
        "X: Exit.\n")
    if request_next in ["x", "X"]:
        sys.exit("Exiting...")
    if request_next in ["a", "A"]:
        get_mapped_proteome(mapping_file_list)

    print("\nChoose from the list, please.")

if __name__ == "__main__":
    sys.exit(main())

```

APPENDIX III

Additional data tables for Chapter 2

Table III-A. Average conservation of loci and orthologous groups across numerous bacterial species.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-A.xlsx>

Table III-B. Average conservation of orthologous groups among protein complex components.

Species	Hu et al. <i>E. coli</i> complexes	EcoCyc <i>E. coli</i> complexes	Kühner et al. <i>Mycoplasma pneumoniae</i> complexes
<i>Mycoplasma genitalium</i> G37	87.98%	87.36%	75.57%
<i>Mycoplasma pneumoniae</i> M129	86.00%	86.26%	61.38%
<i>Helicobacter pylori</i> 26695	79.10%	76.08%	91.19%
<i>Streptococcus sanguinis</i> SK36	78.56%	76.06%	83.12%
<i>Caulobacter crescentus</i> NA1000	75.08%	70.06%	85.87%
<i>Bacillus subtilis</i> subsp. <i>subtilis</i> 168	74.02%	71.85%	82.46%
<i>Escherichia coli</i> K-12 W3110	57.20%	56.01%	85.93%
<i>Pseudomonas aeruginosa</i> UCBPP-PA14	67.76%	65.90%	87.04%

Table III-C. Conservation of orthologous groups and protein-coding loci between pairs of model bacterial species.

Table III-C-A. Counts of total orthologous groups shared between species. Exact strains of each species are identified in the Chapter 2 Methods. Values are locus counts as per shared eggNOG v.3 orthologous groups. Identity values are total OG counts for the given species.

	<i>M. pneumoniae</i>	<i>M. genitalium</i>	<i>B. subtilis</i>	<i>S. sanguinis</i>	<i>H. pylori</i>	<i>C. crescentus</i>	<i>P. aeruginosa</i>	<i>E. coli</i>
<i>M. pneumoniae</i>	461	0.6972	0.0016	0.5707	0.4326	0.4908	0.5324	0.5408
<i>M. genitalium</i>	0.8693	435	0.7116	0.6992	0.5332	0.6100	0.6535	0.6556
<i>B. subtilis</i>	0.0856	0.0846	2582	0.2421	0.1714	0.2564	0.3052	0.3023
<i>S. sanguinis</i>	0.1681	0.1652	0.4814	1437	0.2549	0.3652	0.4255	0.4338
<i>H. pylori</i>	0.1775	0.1754	0.4744	0.3549	1180	0.4894	0.5379	0.5365
<i>C. crescentus</i>	0.0818	0.0815	0.2882	0.2065	0.1987	2231	0.3853	0.3489
<i>P. aeruginosa</i>	0.0554	0.0546	0.2144	0.1503	0.1365	0.2407	3112	0.2884
<i>E. coli</i>	0.0784	0.0762	0.2958	0.2135	0.1896	0.3037	0.4017	2593

Table III-C-B. Percentage of total loci shared between species. Exact strains of each species are identified in the Methods. Values are locus counts as per shared eggNOG v.3 orthologous groups; following values in parentheses are total conservation fractions, e.g., 466 of 482 *M. genitalium* loci appear to be conserved in the *M. pneumoniae* genome (or a fraction of 0.9668) while 517 of 601 *M. pneumoniae* loci appear to be conserved in the *M. genitalium* genome (or a fraction of 0.8602).

	<i>M. pneumoniae</i>	<i>M. genitalium</i>	<i>B. subtilis</i>	<i>S. sanguinis</i>	<i>H. pylori</i>	<i>C. crescentus</i>	<i>P. aeruginosa</i>	<i>E. coli</i>
<i>M. pneumoniae</i>	601 (1)	517 (0.8602)	430 (0.7155)	427 (0.7105)	336 (0.5591)	379 (0.6306)	395 (0.6572)	408 (0.6789)
<i>M. genitalium</i>	466 (0.9668)	482 (1)	388 (0.805)	381 (0.7905)	290 (0.6017)	333 (0.6909)	353 (0.7324)	355 (0.7365)
<i>B. subtilis</i>	644 (0.1588)	627 (0.1546)	4056 (1)	1974 (0.4867)	1374 (0.3388)	2101 (0.518)	2465 (0.6077)	2428 (0.5986)
<i>S. sanguinis</i>	538 (0.2637)	519 (0.2544)	1514 (0.7422)	2040 (1)	804 (0.3941)	1186 (0.5814)	1341 (0.6574)	1379 (0.676)
<i>H. pylori</i>	314 (0.2143)	304 (0.2075)	844 (0.5761)	631 (0.4307)	1465 (1)	916 (0.6253)	979 (0.6683)	992 (0.6771)
<i>C. crescentus</i>	454 (0.1258)	440 (0.122)	2064 (0.5721)	1487 (0.4121)	1401 (0.3883)	3608 (1)	2618 (0.7256)	2414 (0.6691)
<i>P. aeruginosa</i>	678 (0.1174)	655 (0.1134)	3275 (0.5672)	2353 (0.4075)	1963 (0.34)	3615 (0.6261)	5774 (1)	4007 (0.694)
<i>E. coli</i>	594 (0.1433)	567 (0.1368)	2418 (0.5834)	1791 (0.4321)	1414 (0.3411)	2409 (0.5812)	3027 (0.7303)	4145 (1)

Table III-D. Key to short protein complex IDs.

Short ID	EcoCyc ID	Complex Name	Note
C1	F-1-CPLX	ATP synthase F1 complex	
C2	ATPSYN-CPLX	ATP synthase / thiamin triphosphate synthase	
C3	UVRABC-CPLX	UvrABC Nucleotide Excision Repair Complex	
C4	RNAPS-CPLX	RNA polymerase sigma S	
C5	RNAP32-CPLX	RNA polymerase sigma 32	
C6	RNAP70-CPLX	RNA polymerase sigma 70	
C7	APORNAP-CPLX	RNA polymerase, core enzyme	
C8	F-O-CPLX	ATP synthase F0 complex	
C9	HSP70-CPLX	DnaK-DnaJ-GrpE chaperone system	
C10	CPLX0-3801	DNA polymerase III, preinitiation complex	
C11	RIBONUCLEOSIDE-DIP-REDUCTII-CPLX	ribonucleoside-diphosphate reductase 2	
C12	CPLX0-7609	5-carboxymethylaminomethyluridine-tRNA synthase	
C13	PHES-CPLX	phenylalanyl-tRNA synthetase	
C14	CPLX0-7879	NusB-NusE complex	
C15	CPLX0-2424	topoisomerase IV	
C16	CPLX0-2425	DNA gyrase	
C17	PC00027	IHF DNA-binding transcriptional dual regulator	
C18	CPLX0-3934	GroEL-GroES chaperonin complex	
C19	CPLX0-2021	HU DNA-binding transcriptional dual regulator	
C20	RIBONUCLEOSIDE-DIP-REDUCTI-CPLX	ribonucleoside diphosphate reductase 1	
C21	CPLX0-3956	50S ribosomal protein complex L8	
C22	CPLX0-3964	ribosome	
C23	SECE-G-Y-CPLX	SecYEG translocase	
C24	RNAPE-CPLX	RNA polymerase sigma 24	
C25	CPLX0-221	RNA polymerase sigma 19	

C26	RNAP54-CPLX	RNA polymerase sigma 54
C27	CPLX0-222	RNA polymerase sigma 28
C28	ABC-22-CPLX	peptide ABC transporter OppABCDF murein tripeptide ABC transporter
C29	CPLX0-3970	OppBCDFMppA
C30	ABC-8-CPLX	dipeptide ABC transporter
C31	ABC-49-CPLX	glutathione ABC transporter
C32	ABC-20-CPLX	nickel ABC transporter YddO/YddP/YddQ/YddR/YddS ABC transporter
C33	ABC-59-CPLX	glycine cleavage system
C34	GCVMULTI-CPLX	ferric enterobactin transport system
C35	CPLX0-1944	phosphonate ABC transporter
C36	ABC-23-CPLX	CcmEFGH holocytochrome <i>c</i> synthetase
C37	CPLX0-3323	3-phenylpropionate dioxygenase system
C38	HCAMULTI-CPLX	iron (III) hydroxamate ABC transporter
C39	ABC-11-CPLX	ClpAXP
C40	CPLX0-3108	ClpAP
C41	CPLX0-3104	YadG/YadH ABC transporter
C42	ABC-21-CPLX	YbhF/YbhR/YbhS ABC transporter
C43	ABC-57-CPLX	SecD-SecF-YajC-YidC Secretion Complex
C44	SECD-SECF-YAJC-YIDC-CPLX	ferric dicitrate ABC transporter
C45	ABC-9-CPLX	HflB, integral membrane ATP-dependent zinc metallopeptidase
C46	CPLX0-2982	resolvasome
C47	RUVABC-CPLX	Sec Translocation Complex
C48	SEC-SECRETION-CPLX	EntS-TolC Enterobactin Efflux Transport System
C49	CPLX0-5	MacAB-TolC macrolide efflux transport system
C50	TRANS-200-CPLX	degradosome
C51	CPLX0-2381	EmrKY-TolC multidrug efflux transport
C52	CPLX0-2161	

		system	
C53	CPLX0-2121	EmrAB-TolC multidrug efflux transport system	
C54	ABC-62-CPLX	LoICDE ABC lipoprotein transporter	
C55	CPLX0-1981	ferrichrome transport system	
C56	CPLX0-7954	ferric coprogen transport system	
C57	CPLX0-2001	ferric dicitrate transport system	
C58	CPLX0-7934	FtsLBQ cell division complex	
C59	ABC-10-CPLX	ferric enterobactin ABC transporter	
C60	CPLX0-3803	DNA polymerase III, holoenzyme	
C61	CPLX0-2361	DNA polymerase III, core enzyme	
C62	CPLX0-3922	primosome	
C63	CPLX0-241	tagatose-1,6-bisphosphate aldolase 2	
C64	CPLX0-240	tagatose-1,6-bisphosphate aldolase 1	
C65	ABC-25-CPLX	putrescine ABC transporter	
C66	ABC-51-CPLX	YdcS/YdcT/YdcV/YdcU ABC transporter	
C67	ABC-24-CPLX	putrescine / spermidine ABC transporter	
C68	CPLX-158	fructose PTS permease	
C69	CPLX-159	PTS permease - unknown specificity	
C70	CPLX-157	glucose PTS permease	
C71	CPLX0-7	N-acetylmuramic acid PTS permease	
C72	CPLX-164	maltose / glucose PTS permease	Entry removed from EcoCyc but present in MetaCyc
C73	CPLX-168	trehalose PTS permease	
C74	ABC-27-CPLX	phosphate ABC transporter	
C75	PYRUVATEDEH-CPLX	pyruvate dehydrogenase	
C76	2OXOGLUTARATEDEH-CPLX	2-oxoglutarate dehydrogenase complex	
C77	CPLX0-3925	DNA polymerase V	
C78	CPLX-156	mannitol PTS permease - cryptic	
C79	MONOMER0-1761	MalT-MalK	
C80	ABC-16-CPLX	maltose ABC transporter	
C81	CPLX-160	PTS permease - unknown specificity	

C82	ABC-34-CPLX	glycerol-3-phosphate /	
C83	ABC-55-CPLX	glycerophosphodiester ABC transporter	
C84	CPLX0-3958	YcjN/YcjO/YcjP ABC transporter	
C85	EIISGA	EcoKI restriction-modification system	
C86	EIISGC	L-ascorbate PTS permease	
C87	CPLX0-231	PTS permease - unknown specificity	
C88	SUCC-DEHASE	galactitol PTS permease	
C89	CPLX0-3933	succinate dehydrogenase	
		Outer Membrane Protein Assembly	
		Complex	

Table III-E. Conservation of *E. coli* complexes from Hu et al. (2009).

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-E.xlsx>

Table III-F. Essentiality of *E. coli* complexes from Hu et al. (2009).

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-F.xlsx>

Table III-G. Conservation of *E. coli* complexes from EcoCyc.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-G.xlsx>

Table III-H. Essentiality of *E. coli* complexes from EcoCyc.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-H.xlsx>

Table III-I. Conservation of *Mycoplasma pneumoniae* complexes from Kühner et al. (2009).

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-I.xlsx>

Table III-J. Essentiality of *Mycoplasma pneumoniae* complexes from Kühner et al. (2009).

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-J.xlsx>

Table III-K. Experimental protein complexes containing uncharacterized components.

In this table, **Data Set** identifies the source of the complex.

CplxID is the complex identifier assigned by the data source. For the experimentally-observed complexes (Hu et al. (2009) and Kühner et al (2009)) this is the CplxID or Purification ID, respectively.

Size is the number of unique protein components (by OG) in the complex.

Unknown Components is the total number of components in this complex with unknown or unclear functions beyond their membership in the complex. The corresponding OGs must have functional category labels of R or S.

Highly Conserved Components are those coded for by the genomes of at least half of the eight species in the focused set.

Highly Essential Components are those coded for by essential genes present in at least half of the eight species in the focused set.

Data Set	CplxID	Size	Unknown Components	Highly Conserved Components	Highly Essential Components	Notes
Hu et al. (2009)	1	39	13	26	4	degradosome
	2	25	10	15	2	resolvasome
	5	16	7	9	2	
	23	9	6	5	0	HflB complex?
	3	22	4	15	6	ClpP protease complex?
	9	13	4	10	2	efflux transport system?
	15	11	4	7	1	
	4	18	3	13	3	ABC transporter
	11	12	3	9	1	DosC-DosP complex?
	25	9	3	4	1	ABC transporter
	52	6	3	4	1	

	59	6	3	4	1
	105	5	3	4	1
	66	6	3	3	1
	101	5	3	3	1
	26	8	3	6	0
	40	7	3	4	0
	79	5	3	3	0
	90	5	3	3	0
	109	5	3	3	0
	62	6	3	1	0
	12	12	2	7	5
	20	10	2	10	4
	36	7	2	5	4
	41	7	2	7	1
	18	11	2	5	1
	53	6	2	5	1
	91	5	2	4	1
	146	4	2	4	1
	75	5	2	3	1
	80	5	2	3	1
	88	5	2	3	1
	143	4	2	3	1
	151	4	2	3	1
	85	5	2	2	1
	14	11	2	6	0
	33	8	2	6	0
	48	7	2	6	0
	34	8	2	5	0
	100	5	2	5	0

	45	7	2	4	0
	49	7	2	4	0
	55	6	2	4	0
	68	6	2	4	0
	103	5	2	4	0
	116	4	2	3	0
	127	4	2	3	0
	145	4	2	3	0
	149	4	2	3	0
	251	3	2	3	0
	123	4	2	2	0
	245	3	2	2	0
	84	5	2	1	0
	134	4	2	1	0
	154	4	2	1	0
	6	13	1	10	5
	10	13	1	10	5
	27	8	1	8	4
	29	8	1	7	4
	28	8	1	6	3
	30	8	1	6	3
	42	7	1	3	3
	22	9	1	8	2
	21	9	1	5	2
	60	6	1	5	2
	58	6	1	4	2
	137	4	1	2	2
	8	13	1	10	1
	19	11	1	9	1

	17	11	1	8	1
	50	6	1	5	1
	95	5	1	5	1
	97	5	1	5	1
	35	8	1	4	1
	98	5	1	4	1
	107	5	1	4	1
	121	4	1	3	1
	82	5	1	2	1
	102	5	1	2	1
	112	4	1	2	1
	165	3	1	2	1
	173	3	1	2	1
	176	3	1	2	1
	192	3	1	2	1
	232	3	1	2	1
	239	3	1	2	1
	346	2	1	2	1
	204	3	1	1	1
	13	12	1	5	0
	39	7	1	5	0
	46	7	1	5	0
	108	5	1	5	0
	44	7	1	4	0
	54	6	1	4	0
	56	6	1	4	0
	61	6	1	4	0
	78	5	1	4	0
	86	5	1	4	0

	110	4	1	4	0
	120	4	1	4	0
	161	4	1	4	0
	65	6	1	3	0
	67	6	1	3	0
	69	6	1	3	0
	73	5	1	3	0
	87	5	1	3	0
	89	5	1	3	0
	92	5	1	3	0
	94	5	1	3	0
	114	4	1	3	0
	150	4	1	3	0
	198	3	1	3	0
	210	3	1	3	0
	240	3	1	3	0
	72	5	1	2	0
	93	5	1	2	0
	115	4	1	2	0
	118	4	1	2	0
	148	4	1	2	0
	156	4	1	2	0
	181	3	1	2	0
	222	3	1	2	0
	227	3	1	2	0
	237	3	1	2	0
	247	3	1	2	0
	261	3	1	2	0
	262	3	1	2	0

	264	3	1	2	0	
	289	2	1	2	0	
	290	2	1	2	0	
	339	2	1	2	0	
	354	2	1	2	0	
	364	2	1	2	0	
	365	2	1	2	0	
	403	2	1	2	0	
	43	7	1	1	0	
	74	5	1	1	0	
	126	4	1	1	0	
	162	4	1	1	0	
	203	3	1	1	0	
	228	3	1	1	0	
	230	3	1	1	0	
	246	3	1	1	0	
	248	3	1	1	0	
	259	3	1	1	0	
	426	2	1	1	0	
	99	5	1	0	0	
	178	3	1	0	0	
	258	3	1	0	0	
Kühner et al (2009), table S2	50	62	2	39	33	Ribosome
	10	10	2	8	6	Aminoacyl-tRNA synthetase complex
	36	10	2	8	5	Complex function unknown - contains GroL/GroS
	8	16	2	7	1	Protein chaperone complex Complex function unknown
	19	3	2	3	1	- contains thymidylate

						synthase
	49	28	1	19	12	RNA polymerase complex
	44	13	1	8	5	Pyruvate dehydrogenase complex
	15	10	1	7	4	DNA Recombination complex
	29	10	1	5	3	DNA Pol III core complex
						Carbohydrate transport/metabolism
	46	4	1	3	3	complex - contains enolase
	45	11	1	7	2	
	34	5	1	4	2	
	35	5	1	4	2	
	11	2	1	2	2	
	27	9	1	3	1	
	59	2	1	2	1	
	100	2	1	2	1	
	70	2	1	2	1	
	98	2	1	1	1	
	24	5	1	2	0	
	31	4	1	2	0	
	14	3	1	2	0	
	68	2	1	2	0	
	74	2	1	2	0	
	2	5	1	1	0	
	42	4	1	1	0	
	20	4	1	1	0	
	21	4	1	1	0	
	43	2	1	1	0	
	109	2	1	1	0	
	55	2	1	1	0	

	56	2	1	1	0
	58	2	1	1	0
	76	2	1	1	0

Table III-L. Complex-based protein-protein interactions for *E. coli*.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20III-L.xlsx>

This table includes all protein-protein interactions among *E. coli* proteins (as determined by Rajagopala et al. 2014) involved in heteromeric protein complexes (from the EcoCyc set). Each row corresponds to a single protein-protein interaction between the two proteins defined using Uniprot identifiers (**InteractorA** and **InteractorB**).

Each protein is assigned to an OG (specifically, an eggNOG v.4 bactNOG or highest-level NOG where possible, or otherwise a COG; see **OG_A** and **OG_B**).

GroupA and **GroupB** correspond to the EcoCyc protein complex containing InteractorA and InteractorB, respectively.

Cplx_Int_Predicted indicates interactions predicted to occur in the lack of direct experimental evidence but observed experimentally for other proteins of the same two OGs.

FunctionalityA and **FunctionalityB** are general functional categories referring to complex function (of GroupA and GroupB, respectively) rather than individual protein function. The functional categories are primarily based on EcoCyc annotations but also take the shared functions of the component proteins into account.

APPENDIX IV

Additional data tables for Chapter 3

Table IV-A. Counts of literature citing multiple bacterial interactomes.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20IV-A.xls>

The initial version of this data was produced by Christopher Wimble.

All publications in PubMed Central were searched for citations for at least one of the 11 publications listed below, each of which describes a comprehensive bacterial protein-protein interactome.

PMCID	Pubmed Central identifier of the publication.
DOI	DOI of the publication, if available.
Publication Title	Title of the publication.
Citations	The total count of papers, of those listed below, cited by the specified publication. An “x” denotes a citation to the publication in this row.
Rain 2001	Citation to Rain JC, Selig L, De Reuse H, Battaglia V, Reverdy C, et al. (2001) The protein-protein interaction map of <i>Helicobacter pylori</i> . <i>Nature</i> 409: 211–215.
Parrish 2007	Citation to Parrish JR, Yu J, Liu G, Hines J a, Chan JE, et al. (2007) A proteome-wide protein interaction map for <i>Campylobacter jejuni</i> . <i>Genome Biol</i> 8: R130.
Sato 2007	Citation to Sato S, Shimoda Y, Muraki A, Kohara M, Nakamura Y, et al. (2007) A large-scale protein-protein

interaction analysis in *synechocystis* sp. PCC6803. *DNA Res* 14: 207–216.

- Shimoda 2008 Citation to Shimoda Y, Shinpo S, Kohara M, Nakamura Y, Tabata S, et al. (2008) A large scale analysis of protein-protein interactions in the nitrogen-fixing bacterium *Mesorhizobium loti*. *DNA Res* 15: 3–11.
- Titz 2008 Citation to Titz B, Rajagopala S V., Goll J, Häuser R, McKevitt MT, et al. (2008) The binary protein interactome of *Treponema pallidum* - The syphilis spirochete. *PLoS One* 3: e2292.
- Hu 2009 Citation to Hu P, Janga SC, Babu M, Díaz-Mejía JJ, Butland G, et al. (2009) Global functional atlas of *Escherichia coli* encompassing previously uncharacterized proteins. *PLoS Biol* 7: 0929–0947.
- Kuhner 2009 Citation to Kühner S, van Noort V, Betts MJ, Leo-Macias A, Batisse C, et al. (2009) Proteome organization in a genome-reduced bacterium. *Science* 326: 1235–1240.
- Wang 2010 Citation to Wang Y, Cui T, Zhang C, Yang M, Huang Y, et al. (2010) Global protein-protein interaction network in the human pathogen *Mycobacterium tuberculosis* H37Rv. *J Proteome Res* 9: 6665–6677.
- Cherkasov 2011 Citation to Cherkasov A, Hsing M, Zoraghi R, Foster LJ, See RH, et al. (2011) Mapping the Protein Interaction Network in Methicillin-Resistant *Staphylococcus aureus*. *J Proteome Res* 10: 1139–1150.
- Hauser 2014 Citation to Häuser R, Ceol A, Rajagopala S V, Mosca R, Siszler G, et al. (2014) A second-generation protein-protein interaction network of *Helicobacter pylori*. *Mol Cell Proteomics* 13: 1318–1329.
- Rajagopala 2014 Citation to Rajagopala S V., Sikorski P, Kumar A, Mosca R, Vlasblom J, et al. (2014) The binary protein-protein interaction landscape of *Escherichia coli*. *Nat Biotechnol* 32: 285–290.

Table IV-B. All interactions in the meta-interactome network.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20IV-B.csv.tar.gz>

Interactions are provided in PSI-MI TAB 2.7 format, with the addition of orthologous group identifiers for interactor A and B in the 43rd and 44th columns, respectively. The table does not contain a heading for this reason. The file must be decompressed prior to use.

The PSI-MI TAB 2.7 format depends on PSI-MI controlled vocabularies and is described in detail at: <https://code.google.com/archive/p/psimi/wikis/PsimiTab27Format.wiki>

Table IV-C. All interactions in the consensus meta-interactome network.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20IV-C.xls>

In this table:

InteractorA	The first interactor. Either an eggNOG OG identifier or a Uniprot protein identifier, representative of a single-member OG.
InteractorB	The second interactor. Either an eggNOG OG identifier or a Uniprot protein identifier, representative of a single-member OG. For InteractorA and InteractorB, interactors mapping to multiple OGs include all corresponding OGs, separated by commas. For purposes of this data set, multiple-OG interactors are treated as unique OGs, even if their mappings overlap with other OGs.
InteractionCount	Count of individual PROTEIN interactions contributing to this consensus interaction, as per the meta-interactome.
TaxonCount	Count of different taxons (here, a proxy for species) corresponding to the interaction. Similar taxons have been grouped together where possible, e.g. two different <i>E. coli</i> K-12 strains are just considered <i>E. coli</i> K-12.
Taxons	The taxons corresponding to this interaction.
FuncCatA	Functional category of the first interactor.
DescA	Description of the first interactor.
FuncCatB	Functional category of the second interactor.
DescB	Description of the second interactor.

For all FuncCats and Descriptions, multiple-OG interactors include all annotations, separated by pipe (|) symbols. NA indicates that a functional category or description is not available.

Table IV-D. Conserved interactions of unclear function.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20IV-D.xls>

The format of this table is identical to that of **Table IV-C**. All interactions in this table are OG-OG interactions from the consensus meta-interactome where the interaction has been observed in at least two distinct species and involves at least one OG with a functional category of “S”.

APPENDIX V

Additional data tables for Chapter 4

Table V-A. Interactions between bacteriophage and bacterial proteins.

This table is not included in this document due to size.

It may be retrieved from the following location:

<https://github.com/caufieldjh/dissertation/blob/master/Table%20V-A.xlsx>

All interactions in this table are between a bacterial protein and a bacteriophage protein. Induction of a phenotype (i.e., gpX causes lysis) is insufficient evidence, as are host range studies (e.g., gpX mutations allow infection of Species A but not Species B) even if a specific receptor is named. Protein-focused methods (e.g., yeast two hybrid) are acceptable. Reciprocal interactions are included when available but each interaction appears only once between two unique proteins. If an interaction was found in more than one study, all studies are listed under the Source heading.

Phage_Interactor	The protein name of the phage interactor.
Phage	The name of the bacteriophage source of the phage interactor.
Phage_UPID	The Uniprot entry ID of the phage interactor.
Phage_Alt_ID	An alternate ID for the phage interactor if a Uniprot ID is not available. Otherwise, this is identical to Phage_UPID.
Phage_OG	An eggNOG v.4.5 orthologous group assignment for the phage interactor, if available. Otherwise, this is identical to Phage_Alt_ID. Lack of current OG assignment does not preclude future OG assignment.

Host_Interactor	The protein name of the host interactor.
Host	The species name of the bacterial host and source of the host interactor.
Host_UPID Host_Alt_ID	The Uniprot entry ID of the host interactor. An alternate ID for the host interactor if a Uniprot ID is not available. Otherwise, this is identical to Host_UPID.
Host_OG	An eggNOG v.4.5 orthologous group assignment for the host interactor, if available. Otherwise, this is identical to Host_Alt_ID.
ExpMethod	The experimental method used to observe the interaction, as one of the methods defined by the PSI MI 2.5 methods ontology.
ExpMethodID	The ontology ID for the experimental method. See http://www.ebi.ac.uk/ols/beta/ontologies/mi
InfMethod	"Spoke" if the reported interaction is the product of a spoke expansion model. "-" if otherwise.
Source	The first author and publication year of the source of the reported interaction. Multiple sources may be provided for a single interaction but will have different entries.
SourceID	An NCBI PubMed ID for the source.
Database	The source database, if present in a database of protein interactions, or a review article including a collection of interactions.

Table V-B. Citations for phage-host protein interaction sources.

First Author	PMID	Full Citation
Atanasiu et al., 2002	12235377	Atanasiu, C., Su, T.-J., Sturrock, S. S. & Dryden, D. T. F. Interaction of the ocr gene 0.3 protein of bacteriophage T7 with EcoKI restriction/modification enzyme. <i>Nucleic Acids Res.</i> 30, 3936–44 (2002).
Berkane et al. (2006)	16489764	Berkane, E. et al. Interaction of bacteriophage lambda with its cell surface receptor: an in vitro study of binding of the viral tail protein gpJ to LamB (Maltoporin). <i>Biochemistry</i> 45, 2708–20 (2006).
Blasche et al. (2013)	24049175	Blasche, S., Wuchty, S., Rajagopala, S. V & Uetz, P. The protein interaction network of bacteriophage lambda with its host, <i>Escherichia coli</i> . <i>J. Virol.</i> 87, 12745–55 (2013).
Breyton et al. (2013)	24014030	Breyton, C. et al. Assessing the conformational changes of pb5, the receptor-binding protein of phage T5, upon binding to its <i>Escherichia coli</i> receptor FhuA. <i>J. Biol. Chem.</i> 288, 30763–72 (2013).
Briebe et al. (2004)	15297882	Briebe, L. G. et al. Structural basis for the dual coding potential of 8-oxoguanosine by a high-fidelity DNA polymerase. <i>EMBO J.</i> 23, 3452–61 (2004).
Chen et al. (2002)	11751917	Chen, M. et al. Direct interaction of YidC with the Sec-independent Pf3 coat protein during its membrane protein insertion. <i>J. Biol. Chem.</i> 277, 7670–5 (2002).
Cheng et al. (2004)	15302217	Cheng, X., Wang, W. & Molineux, I. J. F exclusion of bacteriophage T7 occurs at the cell membrane. <i>Virology</i> 326, 340–52 (2004).
Ding et al. (1995)	7578104	Ding, Y., Duda, R. L., Hendrix, R. W. & Rosenberg, J. M. Complexes between chaperonin GroEL and the capsid protein of bacteriophage HK97. <i>Biochemistry</i> 34, 14918–31 (1995).
Doublé et al. (1998)	9440688	Doublé, S., Tabor, S., Long, A. M., Richardson, C. C. & Ellenberger, T. Crystal structure of a bacteriophage T7 DNA replication complex at 2.2 Å resolution. <i>Nature</i> 391, 251–8 (1998).
Dove and Hochschild (2001)	11591686	Dove, S. L. & Hochschild, A. Bacterial two-hybrid analysis of interactions between region 4 of the sigma(70) subunit of RNA polymerase and the transcriptional regulators Rsd from <i>Escherichia coli</i> and AlgQ from <i>Pseudomonas aeruginosa</i> . <i>J. Bacteriol.</i> 183, 6413–21 (2001).
Dutta et al. (2004)	15528277	Dutta, S. et al. Crystal structures of 2-acetylaminofluorene and 2-aminofluorene in complex with T7 DNA polymerase reveal mechanisms of mutagenesis. <i>Proc. Natl. Acad. Sci. U. S. A.</i> 101, 16186–91 (2004).
Fornelos et al. (2015)	26138485	Fornelos, N. et al. Bacteriophage GIL01 gp7 interacts with host LexA repressor to enhance DNA binding and inhibit RecA-mediated auto-cleavage. <i>Nucleic Acids Res.</i> 43, 7315–29 (2015).
Ghosh et al. (2008)	18757858	Ghosh, S., Hamdan, S. M., Cook, T. E. & Richardson, C. C. Interactions of <i>Escherichia coli</i> thioredoxin, the processivity factor, with bacteriophage T7 DNA polymerase and helicase. <i>J. Biol. Chem.</i> 283, 32077–84 (2008).
Hinnerwisch et al. (2005)	15989953	Hinnerwisch, J., Fenton, W. A., Furtak, K. J., Farr, G. W. & Horwich, A. L. Loops in the central channel of ClpA chaperone mediate protein binding, unfolding, and translocation. <i>Cell</i> 121, 1029–41 (2005).
Hood and Berger (2016)	27244442	Hood, I. V & Berger, J. M. Viral hijacking of a replicative helicase loader and its implications for helicase loading control and phage replication. <i>Elife</i> 5, (2016).
Kennaway et al., 2009	19074193	Kennaway, C. K. et al. The structure of M.EcoKI Type I DNA methyltransferase with a DNA mimic antirestriction protein. <i>Nucleic Acids Res.</i> 37, 762–70 (2009).

Klenner and Kuhn (2012)	22179606	Klenner, C. & Kuhn, A. Dynamic disulfide scanning of the membrane-inserting Pf3 coat protein reveals multiple YidC substrate contacts. <i>J. Biol. Chem.</i> 287, 3769–76 (2012).
Klenner et al. (2008)	18996118	Klenner, C., Yuan, J., Dalbey, R. E. & Kuhn, A. The Pf3 coat protein contacts TM1 and TM3 of YidC during membrane biogenesis. <i>FEBS Lett.</i> 582, 3967–72 (2008).
Lambert et al. (2004)	15257291	Lambert, L. J., Wei, Y., Schirf, V., Demeler, B. & Werner, M. H. T4 AsiA blocks DNA recognition by remodeling sigma70 region 4. <i>EMBO J.</i> 23, 2952–62 (2004).
Liu and Richardson (1993)	7680479	Lambert, L. J., Wei, Y., Schirf, V., Demeler, B. & Werner, M. H. T4 AsiA blocks DNA recognition by remodeling sigma70 region 4. <i>EMBO J.</i> 23, 2952–62 (2004).
Mallory et al. (1990)	2165499	Mallory, J. B., Alfano, C. & McMacken, R. Host virus interactions in the initiation of bacteriophage lambda DNA replication. Recruitment of Escherichia coli DnaB helicase by lambda P replication protein. <i>J. Biol. Chem.</i> 265, 13297–307 (1990).
Mariano et al. (2016)	27103053	Mariano, R., Wuchty, S., Vizoso-Pinto, M. G., Häuser, R. & Uetz, P. The interactome of Streptococcus pneumoniae and its bacteriophages show highly specific patterns of interactions among bacteria and their phages. <i>Sci. Rep.</i> 6, 24597 (2016).
McMahon et al. (2005)	16170324	McMahon, S. A. et al. The C-type lectin fold as an evolutionary solution for massive sequence variation. <i>Nat. Struct. Mol. Biol.</i> 12, 886–92 (2005).
Miller et al. (2008)	18532877	Miller, J. L. et al. Selective ligand recognition by a diversity-generating retroelement variable protein. <i>PLoS Biol.</i> 6, e131 (2008).
Muñoz-Espín et al. (2009)	19654094	Muñoz-Espín, D. et al. The actin-like MreB cytoskeleton organizes viral DNA replication in bacteria. <i>Proc. Natl. Acad. Sci. U. S. A.</i> 106, 13347–52 (2009).
Mustard and Little (2000)	10692372	Mustard, J. A. & Little, J. W. Analysis of Escherichia coli RecA interactions with LexA, lambda CI, and UmuD by site-directed mutagenesis of recA. <i>J. Bacteriol.</i> 182, 1659–70 (2000).
Odegrip et al. (2000)	10756017	Odegrip, R., Schoen, S., Haggård-Ljungquist, E., Park, K. & Chattoraj, D. K. The interaction of bacteriophage P2 B protein with Escherichia coli DnaB helicase. <i>J. Virol.</i> 74, 4057–63 (2000).
Pani et al. (2009)	19409394	Pani, B., Ranjan, A. & Sen, R. Interaction surface of bacteriophage P4 protein Psi required for complex formation with the transcription terminator Rho. <i>J. Mol. Biol.</i> 389, 647–60 (2009).
Perrody et al. (2012)	23133404	Perrody, E. et al. A bacteriophage-encoded J-domain protein interacts with the DnaK/Hsp70 chaperone and stabilizes the heat-shock factor σ 32 of Escherichia coli. <i>PLoS Genet.</i> 8, e1003037 (2012).
Putnam et al. (1999)	10080896	Putnam, C. D. et al. Protein mimicry of DNA from crystal structures of the uracil-DNA glycosylase inhibitor protein and its complex with Escherichia coli uracil-DNA glycosylase. <i>J. Mol. Biol.</i> 287, 331–46 (1999).
Qi et al. (2015)	26323881	Qi, D., Alawneh, A. M., Yonesaki, T. & Otsuka, Y. Rapid Degradation of Host mRNAs by Stimulation of RNase E Activity by Srd of Bacteriophage T4. <i>Genetics</i> 201, 977–87 (2015).
Qiao et al. (2008)	18836083	Qiao, X., Sun, Y., Qiao, J. & Mindich, L. The role of host protein YajQ in the temporal control of transcription in bacteriophage Phi6. <i>Proc. Natl. Acad. Sci. U. S. A.</i> 105, 15956–60 (2008).
Saikrishnan et al. (2002)	12136137	Saikrishnan, K. et al. Domain closure and action of uracil DNA glycosylase (UDG): structures of new crystal forms containing the Escherichia coli enzyme and a comparative study of the known structures involving UDG. <i>Acta Crystallogr. D. Biol. Crystallogr.</i> 58, 1269–76 (2002).

São-José et al. (2006)	16481324	São-José, C. et al. The ectodomain of the viral receptor YueB forms a fiber that triggers ejection of bacteriophage SPP1 DNA. <i>J. Biol. Chem.</i> 281, 11464–70 (2006).
Sauer et al. (1981)	7021852	Sauer, B., Ow, D., Ling, L. & Calendar, R. Mutants of satellite bacteriophage P4 that are defective in the suppression of transcriptional polarity. <i>J. Mol. Biol.</i> 145, 29–46 (1981).
Serrano-Heras et al. (2006)	16421108	Serrano-Heras, G., Salas, M. & Bravo, A. A uracil-DNA glycosylase inhibitor encoded by a non-uracil containing viral DNA. <i>J. Biol. Chem.</i> 281, 7068–74 (2006).
Sharma and Chatterji (2008)	18359804	Sharma, U. K. & Chatterji, D. Differential mechanisms of binding of anti-sigma factors Escherichia coli Rsd and bacteriophage T4 AsiA to E. coli RNA polymerase lead to diverse physiological consequences. <i>J. Bacteriol.</i> 190, 3434–43 (2008).
Shen et al. (2004)	15169771	Shen, R., Olcott, M. C., Kim, J., Rajagopal, I. & Mathews, C. K. Escherichia coli nucleoside diphosphate kinase interactions with T4 phage proteins of deoxyribonucleotide synthesis and possible regulatory functions. <i>J. Biol. Chem.</i> 279, 32225–32 (2004).
Shtatland et al. (2000)	11058143	Shtatland, T. et al. Interactions of Escherichia coli RNA with bacteriophage MS2 coat protein: genomic SELEX. <i>Nucleic Acids Res.</i> 28, E93 (2000).
Solteszova et al. (2015)	25463056	Solteszova, B., Halgasova, N. & Bukovska, G. Interaction between phage BFK20 helicase gp41 and its host Brevibacterium flavum primase DnaG. <i>Virus Res.</i> 196, 150–6 (2015).
Stephanou et al. (2009)	19523474	Stephanou, A. S. et al. Dissection of the DNA mimicry of the bacteriophage T7 Ocr protein using chemical modification. <i>J. Mol. Biol.</i> 391, 565–76 (2009).
Van den Bossche et al. (2014)	25185497	Van den Bossche, A. et al. Systematic identification of hypothetical bacteriophage proteins targeting key protein complexes of Pseudomonas aeruginosa. <i>J. Proteome Res.</i> 13, 4446–56 (2014).
Van den Bossche et al. (2016)	27447594	Van den Bossche, A. et al. Structural elucidation of a novel mechanism for the bacteriophage-based inhibition of the RNA degradosome. <i>Elife</i> 5, (2016).
Vinga et al. (2012)	22171743	Vinga, I. et al. Role of bacteriophage SPP1 tail spike protein gp21 on host cell receptor binding and trigger of phage DNA ejection. <i>Mol. Microbiol.</i> 83, 289–303 (2012).
Wagemans et al. (2015)	26594207	Wagemans, J. et al. Antibacterial phage ORFans of Pseudomonas aeruginosa phage LUZ24 reveal a novel MvaT inhibiting protein. <i>Front. Microbiol.</i> 6, 1242 (2015).
Wan et al. (2016)	27169810	Wan, H. et al. Structural insights into the inhibition mechanism of bacterial toxin LsoA by bacteriophage antitoxin Dmd. <i>Mol. Microbiol.</i> 101, 757–69 (2016).
Wang et al. (2008)	18157148	Wang, G. et al. The structure of a DnaB-family replicative helicase and its interactions with primase. <i>Nat. Struct. Mol. Biol.</i> 15, 94–100 (2008).
Zillig et al. (1975)	1101258	Zillig, W. et al. In vivo and in vitro phosphorylation of DNA-dependent RNA polymerase of Escherichia coli by bacteriophage-T7-induced protein kinase. <i>Proc. Natl. Acad. Sci. U. S. A.</i> 72, 2506–10 (1975).