Untangling Genomes from Metagenomes

Vaughn S. Iverson

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

E. Virginia Armbrust, Chair

John A. Baross

Robert M. Morris

Program Authorized to Offer Degree:
School of Oceanography

University of Washington

Abstract

Untangling Genomes from Metagenomes

Vaughn S. Iverson

Chair of the Supervisory Committee:
Professor E. Virginia Armbrust
School of Oceanography

Abundant and evolutionarily diverse populations of microbes dramatically shape the ecology of every marine environment on Earth. While it is straightforward to detect and classify members of these natural communities using molecular methods, the vast majority remain inaccessible to laboratory investigation of their physiology and genetic potential, due to the difficulty of predicting and replicating the conditions they may require for growth. This is doubly true for viruses, where to isolate and study a given virus, a susceptible host organism must first be brought into laboratory culture.

Metagenomics has developed as an alternative approach, revealing genetic information about entire microbial communities at once. With rapid technological improvements placing metagenome-scale DNA sequencing within reach of individual researchers, the challenge has shifted to the analysis and interpretation of the enormous resulting datasets. This work describes the development and use of new computational tools to investigate and assemble two metagenomes sampled from Puget Sound during autumn and spring. Community taxonomic classification and quantification revealed a diverse and changing assemblage of Bacteria and Archaea, including enigmatic members of the uncultured marine *Euryarchaeota*.

Targeted metagenomic assembly produced the first genome representing the marine group II *Euryarchaeota*, describing a motile, photoheterotrophic lifestyle, and clarifying the evolutionary origin of the proteorhodopsin gene found in numerous marine Bacteria. This genome further enabled identification and metagenomic assembly of five genomes describing the first reported non-extremophilic archaeal viruses. These genomes reveal a dramatic evolutionary arms race between the viruses and their hosts over control of essential archaeal protein folding machinery in the euryarchaeal host cell. Whether marine *Euryarchaeota* will ever be isolated into culture remains unknown, but through the use of metagenomic genome assembly, key details about how they survive—and perish—in the world's oceans have been revealed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

CONTIG: Contigs are the main output of a *de novo* assembler. They are composed of overlapping shorter reads combined to form a longer contiguous sequence.

COVERAGE: The number of reads that directly "cover" or align with a given position in a sampled genome or known reference sequence.

DE NOVO ASSEMBLY: The computational process of finding reads with significant sequence overlaps that can be used to produce robust contigs of significant lengths. Quality assembly requires good quality reads with few errors and sufficient coverage to ensure that errors are corrected and that there is little missing sequence.

EXTREMOPHILE: Organisms tolerating or requiring extremes of temperature, pH and/or salinity for growth.

HEURISTICS: Computational "rules-of-thumb" that may not universally apply, but which are useful in reducing the complexity of computationally hard problems where exhaustive search for the "best" answer is not feasible.

MATE-PAIR: A pair of reads sequenced from the same physical DNA molecule, but with some amount of unknown sequence between them. Ideally the size of this unknown "insert" is constrained to a fairly tight distribution of possible lengths.

MESOPHILE: Organisms that require moderate conditions for growth. This typically refers to temperature, but may also describe neutral pH and/or moderate salinity levels.

METAGENOMICS: DNA sequencing and analysis performed directly on intermixed fragments of genomes sampled from environmental populations of organisms.

N50: A measure of the quality of a *de novo* assembly: 50% of the sequence in an assembly is found in contigs of greater length than the value of N50.

PHYSICAL-COVERAGE: Analogous to coverage, but is calculated as the number of mate-pairs that map over a given genome position, either one of the reads directly, or the

unknown insert between them (meaning that the reads each align with a sequence on either side of the postion being measured).

READ: The unit of data produced by a DNA sequencer, a string nucleotides bases usually with associated per-base quality information. Depending on the sequencing technology used reads may be anywhere from 25 bp - 10 Kbp in length.

READ ALIGNMENT: The computational process of mapping reads to one or more reference sequences. In a typical read alignment there may be billions of reads and millions of reference sequences, so efficiency and specificity are critical.

RIBOSOMAL DNA (RDNA): The genes coding for the RNA that comprises much of the structure of the ribosome, the protein making machinery of the cell. rDNA is useful because it has highly conserved regions that are nearly universal, and highly variable regions that provide information about which specific group of organisms the sequence came from. Using the conserved regions, rDNA is straightforward to amplify and sequence from environmental samples, and large databases of rDNA sequences have been compiled, making it an extremely valuable tool in initially answering "who's there?" when looking at unknown biological samples.

SCAFFOLD: An ordered list of contigs on the same DNA strand, with small intervening gaps of unknown sequence between each neighboring pair of contigs. Mate-pairing information may be used to form scaffolds from contigs.

SYNTENY: When genes in a given region of two genome sequences being compared occur in nearly the same relative order on both genomes. The presence and extent of synteny between two genome sequences is often used a measure of their relatedness, as distantly related sequences typically demonstrate very little if any synteny.

TETRA-NUCLEOTIDES: Counts of all the overlapping sequences of 4 nucleotides (4-mers) in a genome. Statistics based on the occurrence of tetra-nucleotides relative to what would be neutrally predicted based on counts of tri-nucleotides (3-mers). For sufficiently long sequences, these "tetra-nucleotide anomalies" can be used to cluster sequences into bins belonging to the same or closely related organisms.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to the following people for their invaluable assistance in completing this research and dissertation:

Supervisory Committee:

- E. Virginia Armbrust, John Baross, Susan Hautala, Ed Lazowska, Robert Morris, Gabrielle Rocap and Jay Shendure

Laboratory Assistance and Sampling:

- Chris Frazar, Ellen Lin and Rhonda Morales

Technical Assistance:

- Chris Berthiaume, Cedar McKay and David Schruth

Administrative Support and Counsel:

- Rita Peterson

Consultation:

- Tim Hunkapiller, Anitra Ingalls, David Kingsbury, Willm Martens-Habbena, Michaela Parker and David Stahl

Open Source Software:

- Heng Li, Qiong Wang and Daniel Zerbino

DNA Sequencing:

- Life Technologies Inc.

Financial Support:

- The Gordon and Betty Moore Foundation

DEDICATION

To my stepfather Bob Leifson (1919-2014),
who took the time to teach 10 year old me all about electricity.

To my grandfather Ed Duthie (1924-2015),
who demonstrated to everyone he met what it means to have a positive attitude.

And to my father Loren Iverson (1947-2014),
who introduced me to the joys of reading and exploring the natural world.

Chapter 1
INTRODUCTION

Within each liter of seawater, how do a billion individual microbial cells compete for available resources, interact with one another, shape and respond to environmental changes, and ultimately reproduce, or perish?

Microbes[1] dominate the biosphere. They are the most abundant organisms in every ecosystem on Earth, by one estimate globally totaling 5×10^{30} ($\pm10^{30}$) individual cells; a largely invisible population that comprises approximately 50% of all living biomass [162]. Most of this biomass is found in sediments, but marine microbes suspended within and below the photic zone (above / below ~200 m) of the water column are commonly reported in densities on the order of 5×10^{5} and 5×10^{4} cells/ml respectively; which when integrated over the total volume of the ocean yields a population of roughly 10^{29} cells, comprising an estimated 2.2×10^{15} grams of organic carbon [69, 68, 162].

In addition to being the most numerous organisms, microbes also harbor the bulk of the evolutionary and metabolic diversity contained in life on Earth [105], which they use to play a critical role in maintaining the chemical state of the biome through their exclusive participation in many aspects of the biogeochemical cycling of the elements forming the chemistry of life [38]. Using nitrogen as an example: only microbes possess the enzymes catalyzing the energetically expensive reaction converting inert N_2 gas into ammonia (NH_3),

[1] Although the terms 'microbe' and 'microbial' commonly refer to all microscopic organisms, for the sake of brevity in this document they will exclusively refer to prokaryotic cells (Bacteria and Archaea).

a process which, prior to the industrial production of fertilizers, was the dominant source of biologically available on nitrogen on Earth [60].

Microbes also occupy a pivotal position in the global carbon cycle. Marine cyanobacteria are responsible for an estimated 20–30% of global primary production (oxygenic fixation of CO_2 to organic carbon), a process powered by photosynthesis in the sunlit upper 200 m of the water column, using a small fraction of the biomass that would be required by terrestrial plants to accomplish the same productivity [102, 39, 162, 113]. A most astonishing fact about the standing stock of cyanobacteria is how quickly it "turns over" (is consumed and regenerated) relative to terrestrial ecosystems; a process that occurs in the ocean on the order of days instead of years on land [162].

Heavy grazing on microbes feeds the higher trophic levels of the marine food web—from microscopic protists on up to marine mammals and humans—but about 20% of all marine microbes are also killed daily through infection by an estimated ~10^{30} viruses [144]. Grazing and viral lysis of cells releases a large amount of organic carbon directly into the water, where, as the dominant consumers of dissolved organic compounds, microbes provide a key link in the "microbial loop", consuming carbon (and other nutrients, such as fixed nitrogen) that would otherwise be lost to the dissolved fraction, inaccessible to higher trophic levels [114, 7].

Classification and study of microbes has traditionally been done in the laboratory via isolation from the environment into pure culture, description of morphology via microscopy, and physiological experimentation to study details of growth and metabolism [69]. Decades of this type of foundational work with "model organisms" has led to much of our knowledge of microbial metabolisms. In the 1980s, improvements to methods for enumeration of microbes in environmental samples led to the "great plate count anomaly": an observation that in many oligotrophic (nutrient poor) environments, only 0.1–1.0% of apparently viable

cells would subsequently grow on nutrient enriched media [140], with the implication that a "culture bias" was negatively influencing investigation of the full functional diversity of microbial life.

Around the same time, development of early DNA sequencing techniques enabled inference of the evolutionary relationships among organisms via comparisons of universally conserved genes [164]. These inferred relationships (phylogeny) are typically made using the 16S rDNA gene that codes part of the structural RNA of the ribosome, the protein generating machinery of the cell [104, 106]. This approach led to the revolutionary finding that a group organisms previously thought to be "bacteria" were in fact an evolutionarily distinct third domain of life, now known as the domain *Archaea* [164].

A key property of 16S rDNA sequencing is its amenability for use on environmental samples [106, 104], and many phylogenetic surveys of diverse marine microbial populations have been undertaken [1, 10, 30, 44, 43, 51, 95, 147] leading to the discovery of many previously unknown "phylotypes" of marine *Bacteria* and *Archaea*. The enumeration of these unknown groups of organisms catalyzed a key observation: traditional culture-based studies were dramatically under-sampling the microbial diversity of marine environments, leaving unexplored a vast "uncultured microbial majority" [121].

1.1 Exploring the uncultured majority

Initially within this majority were some of the most numerically successful groups of organisms in the ocean: the SAR11 group of cosmopolitan alpha-*Proteobacteria* [100], the *Prochlorococcus* group of tiny photosynthetic cyanobacteria [24], and the deep dwelling "marine group I" crenarchaea (since reclassified as members of the *Thaumarchaeota* [17], a phylum within the *Archaea*) [68]. These groups are all relatively slow-growing microbes specializing in survival in relatively oligotrophic environments, an observation that led to retooled

culturing efforts using media based on natural seawater, in conjunction with methods that excluded faster growing competitors.

Such an approach was used to bring the first strain of *Prochlorococcus marinus* into culture [23]. With later refinements making the "dilution-to-extinction" procedure more efficient [27], a SAR11 strain was also isolated and called *Pelagibacter ubique* [120]. Once brought into culture, strains of *Pelagibacter* and *Prochlorococcus* have become model organisms for studying the oligotrophic marine environment. Each now has multiple representatives with fully sequenced genomes, yielding valuable new insights into special adaptations they possess for the range of ecological niches they occupy, and the genetic diversity and evolutionary forces at play within closely related phylotypes [126, 52].

Despite the exciting progress made in unravelling the mysteries of *Pelagibacter* and *Prochlorococcus*, other abundant marine groups such as the marine group I *Thaumarchaeota*, the SAR86 group of gamma-*Proteobacteria* [147] and the marine group II *Euryarchaeota* [30] remained resistant to cultivation. The next technological boost came with the development of methods to amplify large pieces of DNA from cells sampled from the environment (~100 K base-pair [bp] pieces of genomes) in *Escherichia coli* using cloning vectors such as "fosmids" or Bacterial Artificial Chromosomes (BACs), followed by screening using 16S rDNA probes to select cloned DNA from specific microbial groups of interest for sequencing [14]. This was significant because it gave a glimpse into the genomes of specific uncultured organisms, creating the opportunity for "environmental genomics" [12].

Early genomic studies of uncultured marine organisms focused on Archaea, yielding informative sequences from both groups I and II [141, 14]. However, it was a BAC clone sequenced from a soil sample that revealed the key insight that group I *Thaumarchaeota* possess genes involved in the oxidation of ammonia, a function previously only known from bacteria, and a key component of the nitrogen cycle [149]. This hint led to success in cul-

turing a strain of marine group I *Thaumarchaeota* (species *Nitrosopumilus maritimus*) isolated from an aquarium [74], and ultimately to the sequencing of its genome [156], revealing its ammonia oxidizing, carbon fixing (chemoautotrophic) metabolism.

Other significant findings have been made through BAC sequencing, such as the discovery of proteorhodopsin—a novel membrane-bound protein that harvests energy directly from light—in a SAR86 clone [13]. Identification of proteorhodopsin was later repeated in several other groups of marine Bacteria including in the genome sequence of *Pelagibacter ubique* [52]; and unexpectedly—as a clear example of an inter-domain horizontal gene transfer—in a BAC from marine group II *Euryarchaeota* [40].

As these examples show, BAC sequencing can be informative, but it has one major drawback: genes of interest need to be positioned in the genome within about 100 Kbp of a 16S rDNA gene, which for a genome of 1.0–5.0 Mbp in total size requires a bit of luck. However, with these and many other discoveries, the environmental genomic approach had shown its utility.

1.2 Metagenomics

Completion of most of the DNA sequencing required for the draft assembly of the human genome [154] freed a massive amount of capacity at DNA sequencing centers. The human genome project also validated the utility of the "shotgun" approach to genome assembly—essentially: fragment DNA randomly, sequence fragments, and use computer algorithms to reassemble [139]—as an alternative to the time consuming method of "chromosome walking" along sequences cloned into BAC-like vectors. With these developments, environmental genomics turned much of its focus toward "environmental shotgun sequencing" [155] (also called metagenomics or community genomics) with the goal of revealing useful genomic data from the "uncultured majority" of microbes in environmental samples

[2, 32, 33, 56, 124].

The first major shotgun metagenome to be published was 76 Mbp of sequence from a relatively simple biofilm community growing in acidic mine drainage (AMD) water [153]. Notably, this project assembled two nearly complete genomes (one each from *Bacteria* and *Archaea*) with partial assemblies for three additional microbes, a result attributed to the small number of phylotypes present (six) and a small amount of genetic heterogeneity within each type [153]. This information enabled a detailed reconstruction of major metabolic functions within members of this community, and interactions between members partitioning functions such as carbon and nitrogen fixation [153, 2].

The next metagenome published a massive (at the time) 1.05 Gbp of DNA sequence from samples taken in the photic zone of the Sargasso Sea, resulting in a "parts list" of over one million distinct genes, and sequence from an estimated 1800 microbial species [155]. A somewhat surprising result, given a community of this complexity, was assembly of two complete genomes from just one of the Sargasso Sea samples [155], a result which was subsequently explained as stemming from a heavily contaminated sample [32].

These initial studies launched the emerging field of metagenomics, with the promise to reveal the roles played by members of the "uncultured majority" [33, 121]. Numerous other shotgun metagenomic studies have been undertaken in the following years, including samples from: whale carcasses and farm soil [150], marine microbes along a depth-profile in the oligotrophic North Pacific Ocean [34], activated sludge bioreactors [92], marine viruses from four ocean basins [4], surface water at multiple Global Ocean Sampling (GOS) stations in the Atlantic and Pacific Oceans [130, 167, 163], termite hind-guts [160], and the gut flora of numerous human subjects [50, 151, 118]. These and other studies have collectively identified literally millions of putative genes from a wide variety of environments, building extensive lists of microbial and viral genes deposited in databases such as CAMERA [133].

With the exception of simple or enriched communities [153, 92, 155], metagenomic studies employing random shotgun sequencing have been largely limited to producing massive "parts lists" of genes from sampled environments, with considerable uncertainty about which phylogenetic groups these genes come from, and which of the "parts" are found together in the same microbial cells. The primary reasons for this are: a) complex communities have diverse populations distributed across many minority subpopulations, b) there is often a high degree of genetic heterogeneity within subpopulations and c) the cost of sequencing deeply enough to resolve these issues has been prohibitive [2, 32, 33, 56, 124]. Whole genome assemblies of cultured organisms have been indispensable in improving our understanding of globally significant microbes through the generation of insights about the niches they exploit, and enabling formation of hypotheses and testing them via laboratory, *in situ*, and further environmental genomic studies [126, 52, 156]. The production of genome assemblies from uncultured groups in metagenomic samples is likely to be just as transformative [150, 157].

Other strategies to achieve the goal of producing whole genomes without isolation into culture have been developed, such as the selection and sequencing of fosmid sequences from an enriched environmental population [157], and using whole-genome amplification techniques to produce "single-cell" genomes [166, 82]. These approaches have had some success; but have mostly produced partial genomes from marine samples, may still require a substantial degree of population enrichment to be effective, and do not provide information about the whole microbial community present in a sample. With major abundant groups of marine microbes (e.g. SAR86 and marine group II *Euryarchaeota*) still mysterious, the goal of revealing their ecological roles through genomic analysis remains a top priority [33].

Sanger-chemistry electrophoretic DNA sequencing technologies [132] have steadily improved in throughput and cost per base since their invention. However, the last decade has

seen an explosion of innovation in new massively parallel high-throughput approaches to DNA sequencing [reviewed in: 136, 135, 109]. Although application of these technologies has initially been restricted in scope due to short read lengths (< 100 bp), high base error rates (> 1%), and a lack of flexible software tools [135], they are a textbook example of a "technological discontinuity" [152]; a fundamental technological leap initially limited to a niche market, but steadily improving to eventually eclipse an incumbent technology.

Sanger capillary sequencing, which was used to sequence the human genome [154] and many metagenomic studies published to-date, is now limited to small incremental technological improvements [135]. Next-generation sequencing technologies represent a fundamental shift that are providing "Moore's Law"-like [97] exponential growth in throughput, with a coinciding reduction in cost per base, into the foreseeable future [135, 109]. This effect, when coupled with continued incremental improvements in read-length, error-rates and software innovations, will likely make these the dominant DNA sequencing technologies in the coming years, relegating Sanger capillary sequencing to niche applications [135, 109].

This dissertation embraces this technological shift, assessing the feasibility of using commercially available next-generation DNA sequencing technologies to generate usable shotgun metagenomic sequence data, with the objective of quantitatively characterizing the sampled microbial communities, and assembling draft-quality whole genome sequences of minority members of those communities.

From: Iverson *et al.*, *Science* 335 (6068): 587-590. Reprinted with permission from AAAS.
Modifications from the original publication are noted at chapter's end in section 2.7.

Chapter 2

REVEALING AN UNCULTURED CLASS OF MARINE EURYARCHAEOTA

Ecosystems are shaped by complex communities of mostly unculturable microbes. Metagenomes provide a fragmented view of such communities, but the ecosystem functions of major groups of organisms remain mysterious. To better characterize members of these communities, we developed methods to reconstruct genomes directly from mate-paired short-read metagenomes. We closed a genome representing the as-yet uncultured marine group II *Euryarchaeota*; assembled *de novo* from 1.7% of a metagenome sequenced from surface seawater. The genome describes a motile, photo-heterotrophic cell focused on degradation of protein and lipids, and clarifies the origin of proteorhodopsin. It also demonstrates that high-coverage mate-paired sequence can overcome assembly difficulties caused by inter-strain variation in complex microbial communities, enabling inference of ecosystem functions for uncultured members.

Abundant and diverse populations of microbes shape every marine environment on Earth [67, 8]. Most environmental microbes are uncultured and resistant to laboratory-based physiological and genomic investigations [140, 121]. The importance of this uncultured majority cannot be overstated, because microbial life plays a critical role in the biogeochemical cycling of elements, maintaining the chemical state of the oceans [38]. How these communities may respond to remediate or exacerbate human impacts is unclear because their ecosystem function remains poorly understood.

Glimpses into the genomes of uncultured microbes are provided by environmental genomic approaches such as single-cell genomics [142] and the sequencing of long segments [10 to 60 kilobase pairs (kbp)] of DNA cloned directly from environmental samples [12]. Shotgun metagenomics bypasses manual isolation of individual sequences, and instead yields

Table 2.1: Metagenome sequencing, analysis, alignment and *de novo* assembly statistics.

	October	May	
Metagenome sequence statistics			
Sample collection date	10-Oct-08	9-May-09	
Raw reads	540	631	millions
Raw sequence	27.0	31.5	gigabases
Mate-pair insert length	1078 ± 312	2084 ± 477	bases ± stddev
Reads filtered for alignments			
Filtered reads	322 (59.6%)	301 (47.7%)	millions (% of raw)
Filtered sequence	15.2 (56.3%)	14.5 (46.0%)	gigabases (% of raw)
Mean trimmed read length	47.2	48.1	bases
RDP 16S rDNA alignment			
Reads recruited	97 (0.031%)	76 (0.025%)	thousands (% of filtered)
Information	0.86	0.75	megabits
RefSeq genomes alignment			
Reads recruited	6.2 (1.9%)	6.4 (2.1%)	millions (% of filtered)
Information	91.3	90.5	megabits
GOS assembly alignment			
Reads recruited	32.1 (10.0%)	24.6 (8.2%)	millions (% of filtered)
Information	619	458	megabits
Contig assembly and alignment			
Contigs assembled	589	517	thousands
Contig sequence	153	155	megabases
Contig N50	369	489	bases
Reads recruited	97 (30.1%)	95 (31.6%)	millions (% of filtered)

large catalogs of randomly sampled environmental genes [155, 34, 130, 167]. Connecting these genes to specific taxonomic groups has been problematic except for groups previously sequenced or those with relative abundances exceeding ~20% [157, 131]. The greater sequencing depth provided by next-generation technologies [91] has recently been exploited to partially assemble multiple genomes from cow rumen microbiomes [59]. Several of these assemblies were estimated to exceed 90% completeness, although all originated from taxonomic orders of Bacteria with cultured and sequenced representatives. We used mate-paired next-generation sequencing of two marine metagenomic samples to reveal the genome sequences of organisms constituting small minorities (< 10%) of the total populations, including an uncultured marine group II *Euryarchaeote* genome.

2.1 Metagenome construction and community analysis

We collected cells (< 0.8μm) from surface waters of Puget Sound in October 2008 and May 2009 (see section 4.1 for methods). Each sample harbored communities of marine bacteria and archaea present at ~10^6 cells per ml (Table B.1). Massively parallel sequencing [91] with a Life Technologies SOLiD™ DNA Analyzer (Life Technologies Corporation, Carlsbad, California) produced a total of 58.5 gigabases of 50-base mate-paired reads (Table 2.1).

Mismatch-tolerant alignments with the Global Ocean Sampling (GOS) [130] and RefSeq [117] databases recruited about 10% and 2% of reads respectively (Tables 2.1, B.2, and B.3), highlighting the substantial proportion of previously unsequenced populations in our samples. A complication of using short sequence reads for metagenomic analysis is that many reads align equally well with multiple sequences in a reference database due to sequence conservation and reference redundancy (e.g. uneven taxonomic coverage). To address this, the short-read alignments were analyzed in an information theoretic (bit-scored) framework [134], as different reads carry varying amounts of information (Figure A.1).

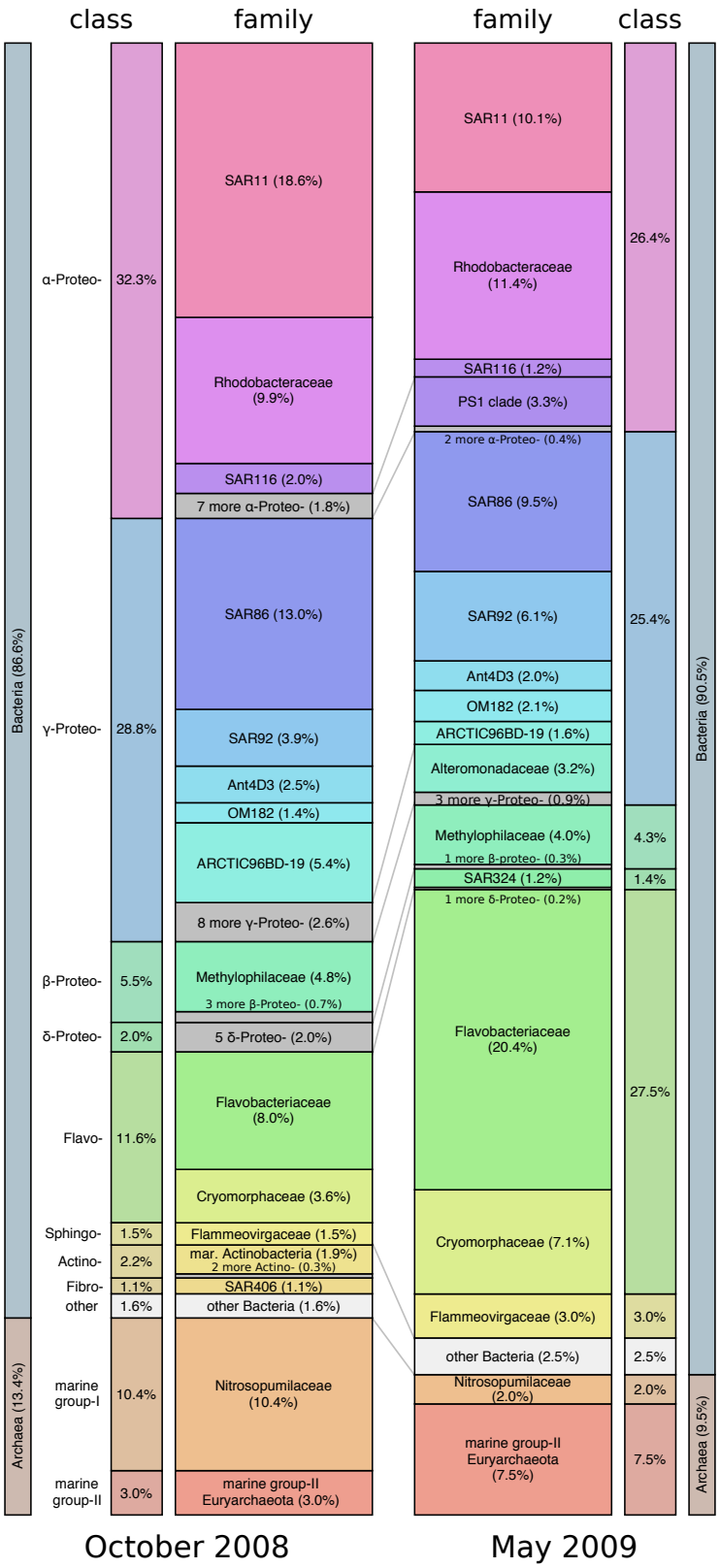

Figure 2.1: Relative abundance of taxonomic groups of Bacteria and Archaea for the October and May samples, based on analysis of metagenomic reads recruited to sequences in the RDP 16S rDNA database. Stacked bars correspond to relative 16S abundance at the domain, class and family taxonomic levels (or approximate equivalent for environmental clades). Colors correspond to the same groups in each sample. White bars represent classes (and their families) that are < 1% relative abundance. Grey bars represent families within a class that are < 1%. Grey lines in the center encompass families collapsed into white or grey bars in one sample or the other.

The sums of bit-scores calculated in this manner are shown for read alignments with each database (Table 2.1). Building on this approach, algorithms were developed to select the most parsimonious reference sequences from each database and to estimate the relative abundance and depth of read coverage for these selected sequences.

The October and May microbial communities were characterized (Figure 2.1) using information theoretic analysis of metagenomic reads aligned to the RDP 16S ribosomal DNA (rDNA) database [26] augmented with full-length 16S rDNA sequences cloned from the October sample (Figure A.2). Database 16S rDNA sequences selected by these analyses were taxonomically classified with a Bayesian method [159] using a custom training set augmented with marine environmental clades, including a previously undescribed group of alpha-*Proteobacteria* dubbed "PS1" (Figures A.3 and A.4).

The 16S rDNA analysis of metagenomic reads detected nearly twice as many family level taxa compared to the 16S rDNA clone sequences (48 versus 28 for the October sample, Figures A.2, A.5 and A.6). Abundance estimates calculated for the identified taxa were validated using simulations that quantified the sensitivity and accuracy of our approach. Simulated populations of 16S rDNA taxa with relative abundance as low as 0.3% were consistently identified and those above 1.0% were typically quantified to within ±10% of their true proportion in the simulated sample (Figure A.7).

Overall, analysis of metagenomic 16S rDNA sequence revealed diverse communities with most family-level taxonomic groups present at less than 10% (Figures 2.1, A.1-A.8). Community composition differed between the October and May samples, with major shifts in population abundances among alpha-*Proteobacteria*, *Flavobacteria* and both phyla of marine *Archaea* (Figures 2.1, A.5 and A.8). Notably, the marine group II *Euryarchaeota* increased in abundance by 2.5-fold between October and May (reaching ~7.5%), a seasonal pattern observed previously [107, 58].

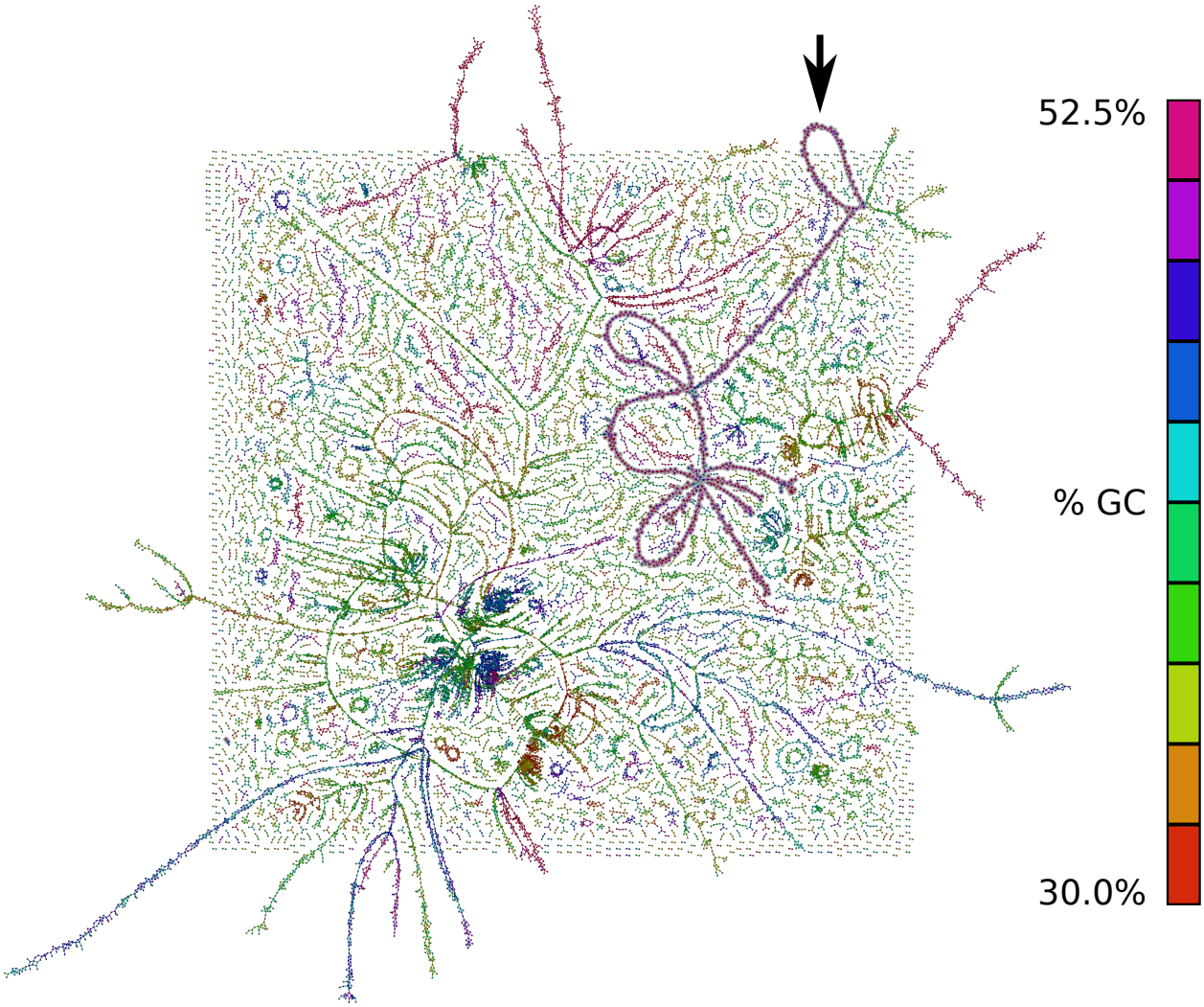

Figure 2.2: Mate-pair connection graph illustrating the May 2009 metagenome *de novo* assembly. Lines represent contigs with mate-pair connections scoring greater than 750 bits (n = 30,945). Long strands represent prokaryote genome sequences, and small circular strands show likely virus or plasmid sequences. Contigs aligning with the MG-II genome assembly are indicated (black arrow, gray shading).

2.2 De novo assembly of genomes from mate-paired metagenomes

De novo assembly of the metagenomic reads produced ~300 megabases of contigs that recruited about threefold more metagenomic reads (~30%) than the GOS database (Tables B.2 and B.3). High-coverage mate-pairing information was exploited to link the assembled contigs (spanning short unassembled sequences), creating metagenomic assembly graphs (Figures 2.2, A.9 and A.10). These connection graphs were split heuristically by using mate-pairing bit-scores, nucleotide composition and read-coverage statistics, producing parsimonious linear scaffolds. Scaffolds were binned by using tetra-nucleotide statistics into candidate genomes, which were taxonomically profiled by using mate-pair connections to informative 16S rDNA regions. Fourteen candidate genomes were produced, each representing 4 to 10% of a sampled community, including representatives of *Euryarchaeota*, *Thaumarchaeota*, *Flavobacteria*, and alpha-, beta- and gamma-*Proteobacteria*. We selected one candidate genome from each sample for more detailed analysis.

A nearly complete genome closely related to the cultured and sequenced *Rhodobacterales* bacterium strain HTCC2255 was reconstructed entirely *de novo* from a population composing ~6.3% of the October sample (see *Thalassobacter*, Figure A.5). Nucleotide alignments of the binned scaffolds to the HTCC2255 reference genome show all 41 scaffolds aligning, with gaps reflecting the lack of metagenomic reads from the October sample recruiting to those regions of the reference genome (Figures A.9 and A.11). The remarkable level of agreement between the assembled and reference genomes demonstrates the robustness of our approach.

A closed genome (designated herein as MG-II) representing the marine group II *Euryarchaeota* was reconstructed via *de novo* assembly of 11 mate-pair connected scaffolds with highly correlated sequence statistics (Figures 2.2 and A.12) from a population composing ~7.5% of the May sample (Figure A.8). A small hyper-variable region (HVR) was insuffi-

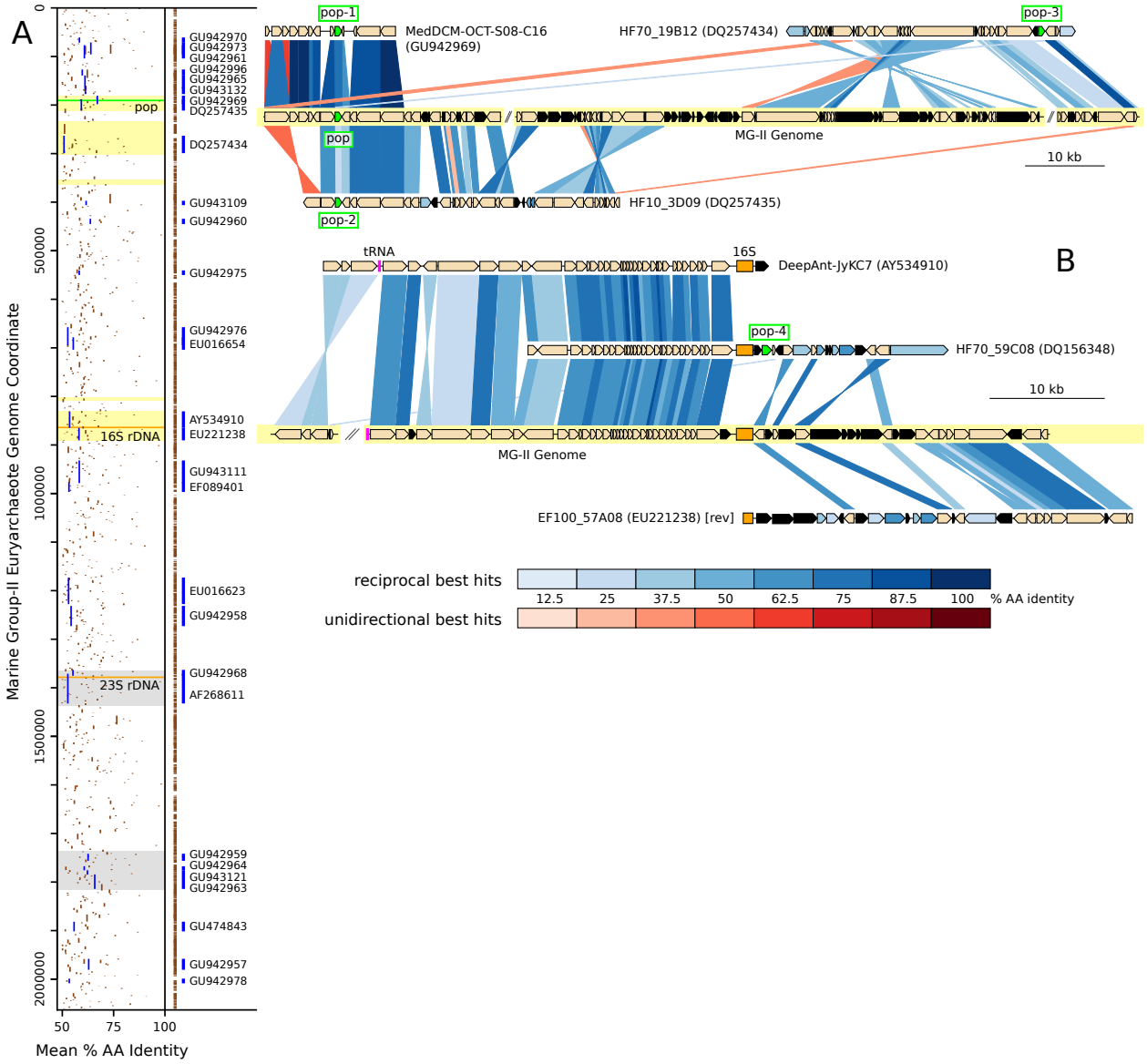

Figure 2.3: **(A)** Alignment of environmental sequences with the MG-II genome. Mean amino acid identity and position indicated for aligned Genbank environmental clones (blue lines) and assembled metagenomic contigs (brown lines). Blue and brown vertical lines (right) summarize tiled coverage in plot area; accession numbers for aligned Genbank sequences are noted. Shaded regions correspond to alignments detailed in (B) (yellow) and Figure A.16 (grey).

(B) Comparison of homologous proteins between environmental clones and two regions of MG-II (yellow backgrounds); breaks between segments indicated by //. Homologous genes connected by shaded regions indicate reciprocal (blue) and unidirectional (red) best hits. Black-shaded gene models (pointed boxes) indicate no blast hits, blue-shaded indicate a reciprocal best hit to an undepicted location in MG-II, and beige otherwise. Genes encoding proteorhodopsin are highlighted in green. Clone identifiers and accession numbers are shown for environmental sequences; those marked [rev] are reversed.

ciently covered for complete automated assembly. Two variants with coverage of ~20× were partially reconstructed, and one of these (~14 Kbp) was finished using Sanger sequencing to fill remaining gaps. The resulting circular chromosome of 2.06 megabases recruited ~5 million aligned reads (1.7% of the May metagenome), yielding 118-fold read coverage and 2194-fold physical coverage by mate-pair inserts (Figures A.13 and A.14). The proportion of coverage of the two HVR variants to the genome average and analysis of 16S rDNA sequences (Figure A.15) each predict that five or more strains of marine group II *Euryarchaeota* were present in the May sample. Our assembly therefore represents a "majority rules" consensus, with possible genomic rearrangements at scaffold boundaries in some strains.

The mesophilic marine group II *Euryarchaeota* have a cosmopolitan distribution and are relatively abundant during summer months [107, 58]. The metabolism of this uncultured group has been mysterious, with only a handful of sequenced environmental clones containing definitive phylogenetic markers (16S and 23S rDNA) available to infer its biogeochemical role (Figure A.15) [14, 98, 40, 94]. To evaluate the correspondence of the MG-II genome to known environmental sequences and identify additional sequences not previously ascribed to this group, we performed six-frame translated alignments of Genbank environmental sequences to the MG-II genome. Sixty-seven environmental clones and 2894 metagenomic assemblies were identified that covered 36% and 96%, respectively, of predicted MG-II protein coding sequences (Figure 2.3A). Clones derived from globally distributed marine samples revealed significant conservation of gene order with the MG-II genome assembly (Figures 2.3B and A.16), reaffirming the group's cosmopolitan distribution.

2.3 Inference of the origin of proteorhodopsin

Of particular interest is the single-copy proteorhodopsin (*pop*) gene present in the MG-II genome, which encodes key active-site residues shared with the light-driven proton-pump-

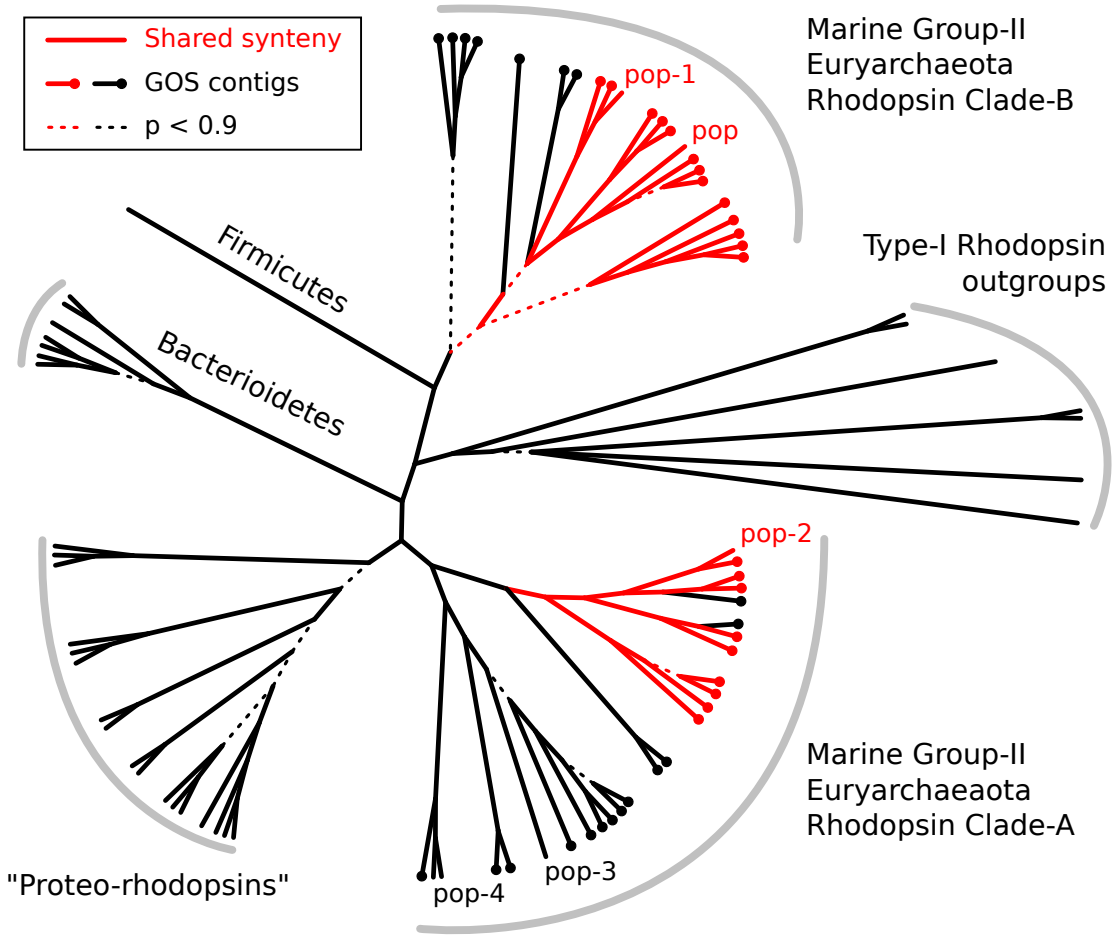

Figure 2.4: Unrooted cladogram depicting Bayesian phylogeny of selected rhodopsin proteins. Major clades are labeled taxonomically, with "Proteo-rhodopsins" denoting those found in marine *Proteobacteria* (genomes and environmental clones). Out groups include bacterial and archaeal "bacteriorhodopsins," archaeal halorhodopsins and sensory rhodopsins. The MG-II rhodopsin (pop), and four proteins from marine group II *Euryarchaeota* environmental clones (pop-1 to pop-4) are identified.

ing rhodopsin found in *Exiguobacterium sibiricum* [108]. Environmental marine group II euryarchaeal *pop* genes appear in at least three MG-II genetic contexts: the position originally detected by Frigaard *et al.* [40] (pop-4, Figure 2.3B); that of the MG-II *pop* gene (pop, Figure 2.3B), confirmed by two environmental clones (pop-1 and pop-2, Figure 2.3B); and that found in an environmental clone aligning with a third region of the MG-II genome ("pop-3", Figure 2.3B). These five *pop* genes cluster phylogenetically into two distinct clades that also contain 42 *pop* genes identified in metagenomic assemblies aligning with positions in the MG-II genome. Clade A shares a common ancestor with proteorhodopsins from *Proteobacteria* [40], whereas clade B is diverged from all known bacterial proteorhodopsins [49] (Figures 2.4 and A.17). The only synteny shared by *pop* genes from both clades ($n = 22$; in red, Figures 2.4 and A.17) is the genetic context seen in the MG-II genome (pop, pop-1, pop-2; Figure 2.3B), indicating that the most parsimonious origin of clade A is a euryarchaeal ancestor shared with clade B. Thus, the proteorhodopsin of marine *Proteobacteria* appears to have originated in marine group II *Euryarchaeota*.

2.4 A motile, photo-heterotroph, degrading protein and lipids

The MG-II genome sequence reveals 1781 predicted proteins, single copies of each of the four ribosomal RNA genes, and a typical number of tRNAs and noncoding RNAs (Table 2.2). The MG-II 16S rDNA gene clusters with the Group II.a clade of marine group II *Euryarchaeota* [45] (Figure A.15). An analysis using 31 conserved archaeal proteins (Tables B.4 and B.5) places the marine group II *Euryarchaeota* in the most deeply rooted position of a clade containing primarily thermoacidophiles, including *Aciduliprofundum boonei* [123] and members of order *Thermoplasmata* (Figure 2.5A). The MG-II genome shares the same compliment of ribosomal proteins found in *A. boonei* with complete conservation of gene order within multi-gene clusters (Table B.6), indicating *A. boonei* is the most closely related sequenced

Table 2.2: Marine group II *Euryarchaeote* genome statistics in comparison with *Aciduliprofundum boonei* and *Nitrosopumilus maritimus*.

	Marine Group II	*A. boonei*	*N. maritimus*
Length	2,063,378	1,486,778	1,645,259
%GC	51.5%	31.2%	34.2%
rRNAs	4	3	4
tRNAs	41	32	40
ncRNAs	2	2	1
Coding genes	1781	1550	1797
Mean length	367	304	277
% coding	95.0%	95.0%	90.6%
Unknown function	577 (32.4%)	444 (28.6%)	713 (39.7%)
Peptidases	52 (2.9%)	41 (2.6%)	20 (1.1%)
Mean length	2151	1665	1170
% of coding	5.7%	4.8%	1.6%
Non-peptidase homologs	14	19	4

organism to the marine group II *Euryarchaeota*.

Core MG-II archaeal metabolic genes include those encoding glycolysis and gluconeogenesis pathways, a complete tricarboxylic acid cycle, and oxidative phosphorylation complexes. Genes dedicated to performing autotrophy or methanogenesis using known pathways are absent. MG-II encodes proteins potentially requiring and transporting cobalamin (vitamin B_{12}) and biotin (vitamin B_7) but does not appear to possess pathways to synthesize these compounds *de novo*. Additionally it does not appear to possess enzymes to reduce nitrate or sulfate, implying a dependence on reduced forms of N and S. An operon containing genes for archaeal flagellar proteins similar in gene order to those identified in *A. boonei* [123] strongly suggests the capacity for motility. We interpret these observations to indicate a motile heterotrophic lifestyle.

Among MG-II genes are many large putative peptidases (Tables B.7 and B.8), which are over-represented in the genome relative to those in the known protein degrader *A. boonei* [123] (5.7% versus 4.8% of coding sequence, Table 2.2). This adds considerable new evidence to previous speculation [14, 98] that consumption of protein may be an important metabolic activity shared in common with the peripherally related *Thermoplasmata* [129]. Unexpectedly, we identified enzymes most similar to bacterial proteins that form a complete fatty acid degradation pathway (Table B.9), suggesting marine group II *Euryarchaeota* may catabolize straight chain lipids. The degradation of protein and lipid suggests that particles may be an important growth substrate, and MG-II codes for proteins with a variety of adhesion domains, as well as type-II/IV secretion systems to transport such proteins to the cell surface.

We identified enzymes necessary to synthesize precursors for archaeal ether-linked lipids and also identified several putative acyl-carrier-protein fatty-acid synthesis enzymes and enzyme homologs for glycerolipid biosynthesis, most similar to bacterial proteins (Table B.9)

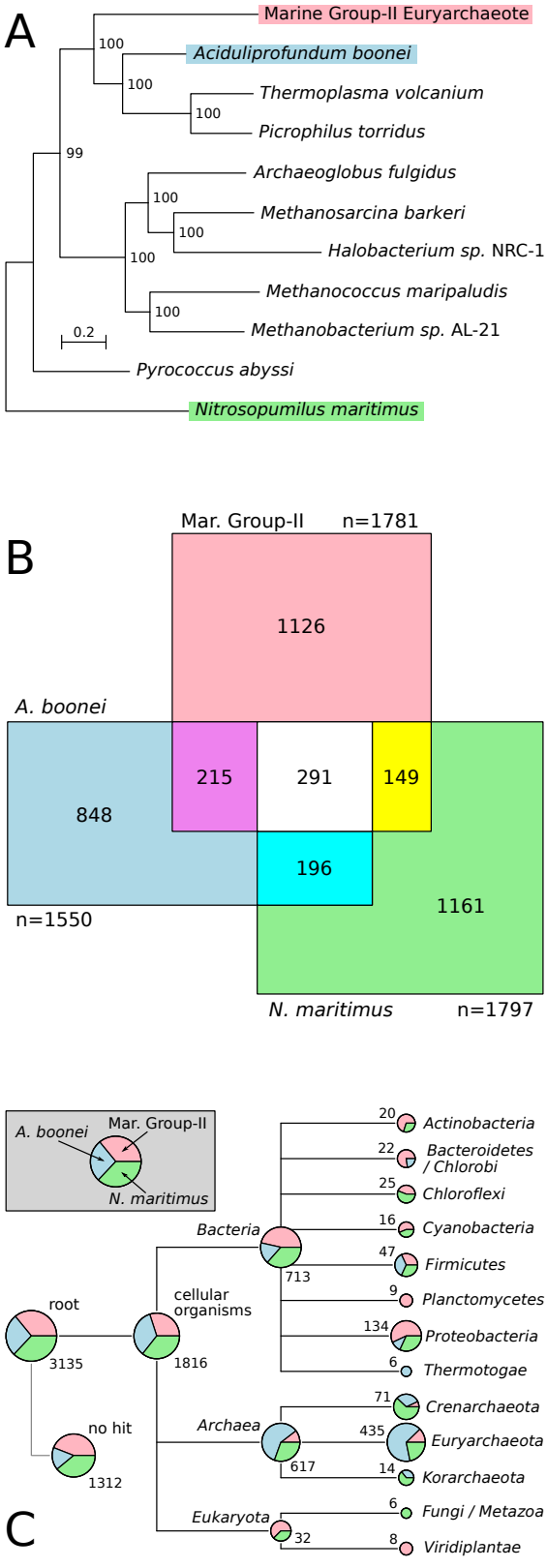

Figure 2.5: Comparison of the marine group II *Euryarchaeote* genome (MG-II) to other sequenced Archaea.

(A) Maximum likelihood phylogeny of ten selected *Euryarchaeota* with available genomes, based on the concatenated amino acid sequences of 31 conserved archaeal proteins. The tree is rooted with *Nitrosopumilus maritimus* (`NC_010085`), a member of the *Thaumarchaeota*. Branch lengths are proportional to the number of substitutions per site. Bootstraps are shown for 100 replicate trees. The genomes of *N. maritimus* and *Aciduliprofundum boone*i (`NC_013926`) are selected for comparison with MG-II.

(B) Venn diagram depicting protein homologs shared among *A. boonei*, *N. maritimus* and MG-II. Homologous proteins were defined by reciprocal blast hits (RBHs) with an e-value cutoff of 10^{-5}. Areas are proportional to the number of proteins indicated.

(C) Taxonomic assignment of proteins without shared homologs from *A. boonei*, *N. maritimus* and MG-II, based on classification of blast hits to a RefSeq protein database. The tree represents NCBI taxonomic hierarchy, to the level of phyla. The pie chart at each node represents the number of proteins from each genome assigned to that level, with total pie area scaling as the log of the (noted) total number of proteins in each pie. The "root" node represents all non-homologous proteins in the comparison genomes. The "no hits" node represents proteins without blast hits to the database. All other nodes represent the number of proteins classifiable to that taxonomic level or below (Table B.11).

and without homologs among the sequenced Archaea. The exclusive use of ether-linked isoprenoid lipids is considered a signature trait of nearly all Archaea [73], so the possible production and function of Bacteria-like ester-linked lipids requires further examination.

2.5 Many MG-II proteins show a bacterial signature

We performed an initial analysis of gene homologs shared among seven selected archaeal genomes (Table B.10), and selected *A. boonei* (a marine thermoacidophile and the genome with the largest number of shared homologs, $n = 506$), and *Nitrosopumilus maritimus* [156] (the only other free-living marine archaeon with a finished genome) for detailed comparison. Homologous genes shared among these three genomes (Figure 2.5B), reveal a small shared core ($n = 291$) primarily composed of characteristically archaeal proteins required for replication, transcription, translation and core metabolic functions.

Curiously, over 60% of predicted MG-II proteins have no homologs in these other archaeal genomes, with many having no significant homology to any RefSeq genome ($n = 578$). To explore potential relationships of the "non-shared" proteins to homologs that were found in RefSeq genomes, taxonomic assignments were calculated for the non-homologous proteins among the three comparison genomes (Figures 2.5C, A.18 and Table B.11). About 50% of the "non-shared" MG-II proteins had homologs in the RefSeq database, with 60% of those ($n = 332$, 18.6% of total) assigned to bacterial homologs, which is striking in comparison to *A. boonei*, where only 19% were similarly assigned ($n = 119$, 7.7% of total, Table B.11). The opposite was also observed, with 59% ($n = 366$, 23.6% of total) of *A. boonei* "non-shared" proteins with RefSeq homologs assigned to Archaea, whereas only 11% ($n = 64$, 3.6% of total) of equivalent MG-II proteins were so assigned (Table B.11). Overall, the genes with bacterial homologs (including those for ester-linked lipid metabolism) are distributed across all of the well-supported scaffolds in the MG-II assembly (Figure A.14), suggesting marine group

II *Euryarchaeota* share many genetic adaptations with Bacteria.

2.6 Concluding remarks

Marine group II *Euryarchaeota* are likely motile photo-heterotrophs focused on protein and lipid degradation. Proteorhodopsin, which appears to be a euryarchaeal innovation, may provide beneficial supplemental energy for propulsion to cells traversing relatively large distances in search of food particles [158]. *De novo* assembly and analysis of the MG-II genome from high-coverage mate-paired metagenomic sequence demonstrates the power of this approach to provide insights into the roles uncultured microbes play in a changing global environment.

2.7 Modifications from original publication

The following modifications were made to the text of this chapter relative to the content originally published in [64], and are identified here in compliance with the AAAS conditions for reprinting material previously published in *Science* within an electronic dissertation. Every effort was made to preserve the content and meaning of the original publication while adapting it to conform with the style and organization of this dissertation.

- The Supporting Online Text from the Supporting Online Material (SOM) of [64] has been integrated into the chapter text verbatim, replacing a small amount of summarizing text from the original. These changes occur at two locations in the chapter text, corresponding with the two named sections from the original SOM text. A small amount of new text is added around the inserted SOM sections to preserve continuity with the surrounding original text.
- Materials and methods from the SOM have been incorporated into chapter 4
- The text is now divided into sections named to summarize the main topic of each.

- Two figures and two tables from the SOM have been moved into the body of the chapter text.
- The remainder of the SOM figures and tables are now in Appendix A and Appendix B respectively.
- Table and figure references have been renumbered to conform with the style of this dissertation, and to reflect the above changes.
- Citations have been changed and renumbered to conform with the style of this dissertation, and references are integrated into the full Bibliography.
- Acknowledgments have been incorporated into those of this dissertation.

Chapter 3

GENOMES OF THE FIRST MESOPHILIC ARCHAEAL VIRUSES

Viruses are the most numerous predators on Earth and significantly impact the ecology of their largest habitat, the oceans. Thousands of bacterial viruses have been identified, but only around one hundred archaeal viruses have been described; all of which infect extremophiles. Here we report five marine euryarchaeal virus genomes, representing the first known non-extremophilic archaeal viruses. They form three groups with circular genomes of ~65, 105 and 130 Kbp, assembled from metagenomes we previously used to construct the first mesophilic marine *Euryarchaeote* genome. All three viral groups possess signature euryarchaeal DNA processing genes, and the smaller two are related and encode viral head-tail genes. Interestingly, each genome contains a gene encoding an HSP60-family protein chaperonin subunit, either bacterial or archaeal, the latter of which appear to have provoked an evolutionary arms race for control of the archaeal thermosome protein folding complex of marine group II *Euryarchaeota.*

All organisms are hypothesized to be vulnerable to viral infection, and in the marine environment virus particles typically outnumber all living organisms by a factor of 10 [144, 15]. Despite their small nanometer-scale size, viruses are the second highest pool of biomass in the sea [144]. Most of these viruses infect microscopic, unicellular hosts and are responsible for an estimated 10-40% of the mortality of the primary producers and heterotrophic scavengers of organic matter in the ocean [41]. This disrupts the traditional notion of the "food-web" by diverting organic carbon away from higher trophic levels, shunting it back into particulate and dissolved matter to be consumed again by heterotrophic Bacteria and Archaea [165].

Dozens of viruses have been isolated for important cultured groups of marine Bacteria and Eukaryotes [169, 143, 61, 127, 101, 148, 21]. Despite the ubiquity and importance of

mesophilic Archaea [31], the only known marine archaeal virus comes from the extreme environment of a deep-sea hydrothermal vent [48]. Of the approximately 100 archaeal viruses that have been isolated and sequenced, all infect extremophilic hosts, including hyperthermophiles, halophiles and methanogens [138, 115]. These archaeal viruses are either single or double stranded DNA (dsDNA) viruses with linear or circular genomes and many share common head-tail morphologies with well-studied bacteriophage groups [111]. The potential magnitude of the unexplored diversity within archaeal viruses is underscored by the fact that more novel morphotypes of viruses are already known from Archaea than from the entire—much better studied—bacterial domain [115]. Because mesophilic Archaea dwell in soil and the oceans, their undescribed viruses have great potential to yield significant new discoveries about these abundant members of Earth's largest ecosystems [31, 68, 22, 138].

Two main groups of mesophilic marine Archaea are known: the autotrophic ammonia-oxidizing members of the *Thaumarchaeota* [17]; and the marine *Euryarchaeota* belonging to monophyletic 16S rDNA clades designated as groups II and III [44, 42]. Cultured representatives are not available for either group of marine *Euryarchaeota* and insight into the physiology of these organisms is derived primarily from culture independent methods such as targeted sequencing of large-insert fosmid clone-libraries [93] and assembly of deeply sequenced next-generation metagenomes [64]. A distinctive feature of marine *Euryarcheote* genomes is the abundance of horizontally transferred bacterial genes, suggesting that bacteria were a reservoir of genetic diversity facilitating adaptation to the mesophilic marine ecosystem from a primordial extremeophilic ancestor [36, 64].

In the absence of a suitable cultured host organism, similar culture independent methods have been directed at sequencing viral genomes with recent studies yielding entire small virus genomes based on fosmid clones, single cell genome amplification, and *de novo* metagenomic assembly [96, 79, 137]. In our previous report on the marine group II *Euryarchaeote*

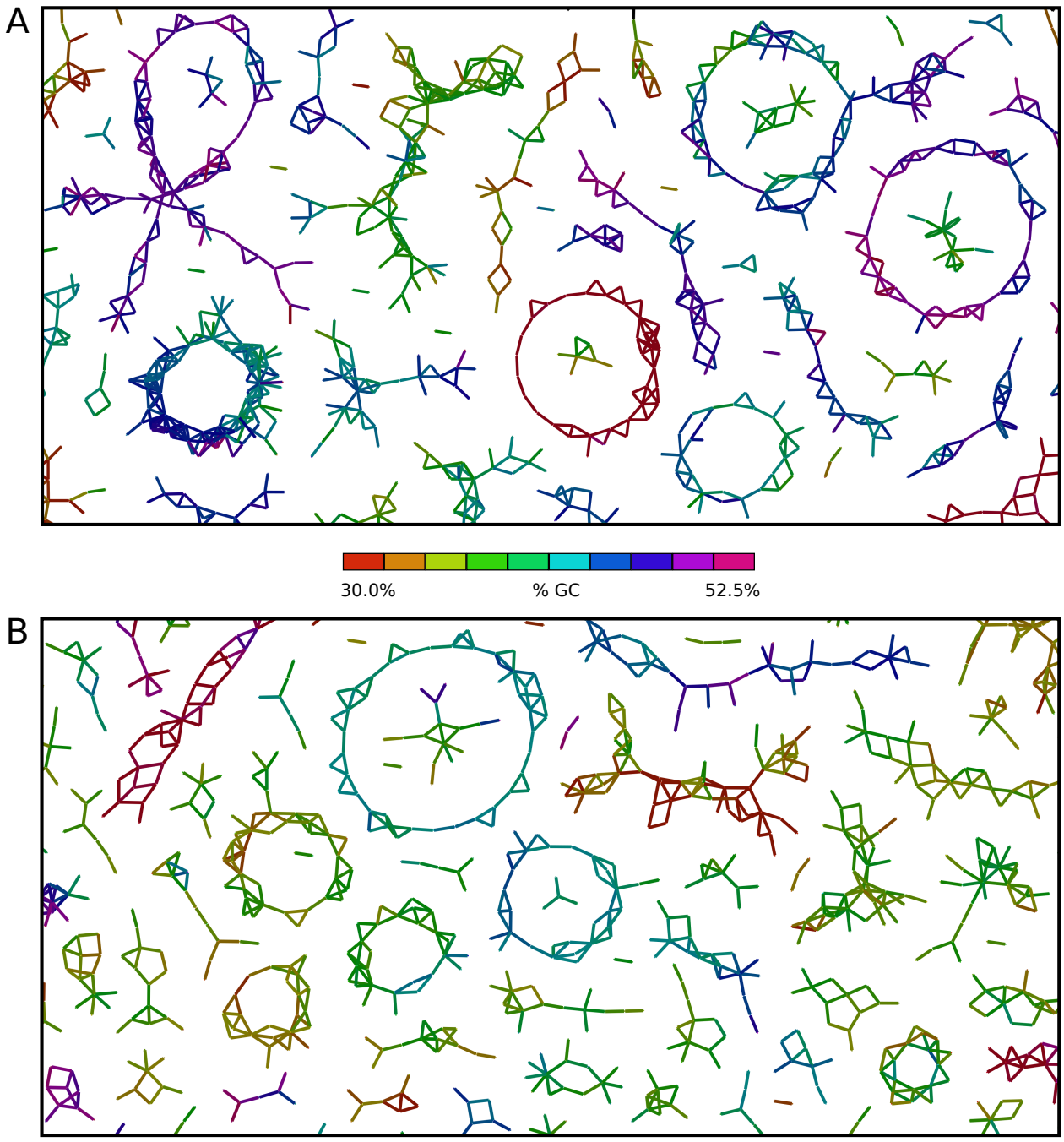

Figure 3.1: Zoomed regions from **(A)** October and **(B)** May mate-pair connection graphs showing small circular putative virus genomes and plasmids. These are analogous to regions of Figures A.9 and A.10, although they depict new *de novo* metagenome assemblies (see methods).

genome (hereafter MG-II), we noted that marine Archaea composed about 10% of the two metagenomes we sampled in October 2008 and May 2009. We also noted many small circles visible in the mate-pair connection graphs, speculating that these likely represent circular virus genomes or plasmid sequences [64] (Figures 2.2 and 3.1). Here we report archaeal virus genomes found among these small circular scaffolds.

3.1 Five euryarchaeal virus genomes

We created multiple new *de novo* assemblies for the October and May metagenomes, varying assembly parameters to produce five alternative contig assemblies for each sample, totaling over two hundred million bases of unique sequence (Table B.12, see section 4.6 for methods). These contigs were then scaffolded using custom software as previously described [64] and the reads mapping to circular scaffolds were segregated and independently reassembled. This resulted in 59 (2.8 Mbp) and 91 (4.8 Mbp) circular scaffolds for the October and May samples, respectively (Figures A.19 and A.20, Tables B.13 and B.14). Within these 150 putative circular viral dsDNA genomes we identified five genomes (hereafter MEV1 - MEV5, Figures A.19 and A.20, Tables B.13 and B.14) that included genes with high scoring BLAST hits to known euryarchaeal and viral sequences. Given the presence of *Thaumarchaeota* in these samples [64], it seems likely that viruses infecting these hosts would also be present (and of interest since there are no known thaumarchaeal viruses [75]). However, thaumarchaeal genes were not detected in any of the circular scaffolds, perhaps because the unknown thaumarchaeal viruses possess genomes that are non-circular, not composed of dsDNA, or which do not carry identifiable genes of archaeal origin.

Table 3.1: Summary statistics for assemblies of archaeal virus genomes (MEV1-MEV5) and additional linear scaffolds. Groups are clustered based on similar genome sizes and homologous proteins conserved among group members. The October sample was collected on 10-Oct-08, the May sample on 9-May-09. Size is the genome/scaffold length in thousands of nucleotides. Coverage denotes fold-coverage of the scaffold by metagenome sequence reads. Annotated indicates the number of genes with predicted functions (percent of total genes). Conserved is the number of genes that have homologs within other genomes or scaffolds in the same group (percent of total genes).

Name	Group	Sample	Size	GC %	Coverage	Genes	Annotated	Conserved
MEV1	1	October	106	47.0%	49×	119	19 (15%)	60 (50%)
MEV2	1	October	105	46.1%	87×	109	18 (16%)	61 (55%)
GG3SC144	1	May	96	37.0%	40×	104	15 (14%)	44 (42%)
MEV3	2	October	65	44.6%	67×	84	19 (22%)	35 (41%)
GG2SC877	2	October	61	47.1%	43×	82	15 (18%)	35 (42%)
MEV4	3	October	129	45.0%	163×	189	22 (11%)	176 (93%)
MEV5	3	May	128	45.0%	62×	187	22 (11%)	176 (94%)
GG2SC16	3	October	120	45.3%	26×	152	16 (10%)	51 (33%)

Annotation and analysis of predicted genes from the MEV genomes revealed that these viruses carry key archaeal proteins with phylogenetic signatures strongly suggesting that they infect marine *Euryarchaeota*. Predicted genes from the MEV genomes were used to search for additional metagenomic scaffolds that were not classified as circular. Three additional scaffolds were identified based on protein homology. The five MEV genomes and three additional scaffolds grouped into three genome size classes (referred to as groups 1–3). Evidence for the relatedness of sequences within each group is based on the number, similarity and conservation of gene order for homologous proteins conserved within each group (Table 3.1 and Figure 3.2).

Two genomes (MEV1, MEV2) of similar size and GC content (~105 Kbp and ~46%) from the October sample were assigned to Group 1 (Table 3.1). A linear scaffold (`GG3_SC144`) from the May metagenome shares significant protein homology and gene order with the genomes, although it differs markedly in GC content (37%). About 50% of the predicted genes in the Group 1 genomes have homologs within other Group 1 sequences (Figure 3.2A and Table 3.1) and about 15% of the genes were assigned non-hypothetical annotations (Figure A.21 and Table B.16).

The single genome (MEV3) assigned to Group 2 has the smallest of the five MEV genomes (65 Kbp). A linear scaffold (`GG2_SC877`) assembled from the same metagenome (October) was also assigned to the group based on protein homology and gene order. About 40% of the MEV3 genes have homologs with `GG2_SC877` (Figure 3.2B and Table 3.1), and about 20% of the genes are assigned non-hypothetical annotations (Figure A.22 and Table B.17).

The two genomes assigned to Group 3 (MEV4, MEV5) were assembled from the October and May samples respectively, and produced large genomes (~130 Kbp) that are highly similar, with remarkably high nucleotide identity of 94% and near 100% amino acid identity for all but a few predicted proteins (Figure 3.2C). The Group 3 genomes represent the only

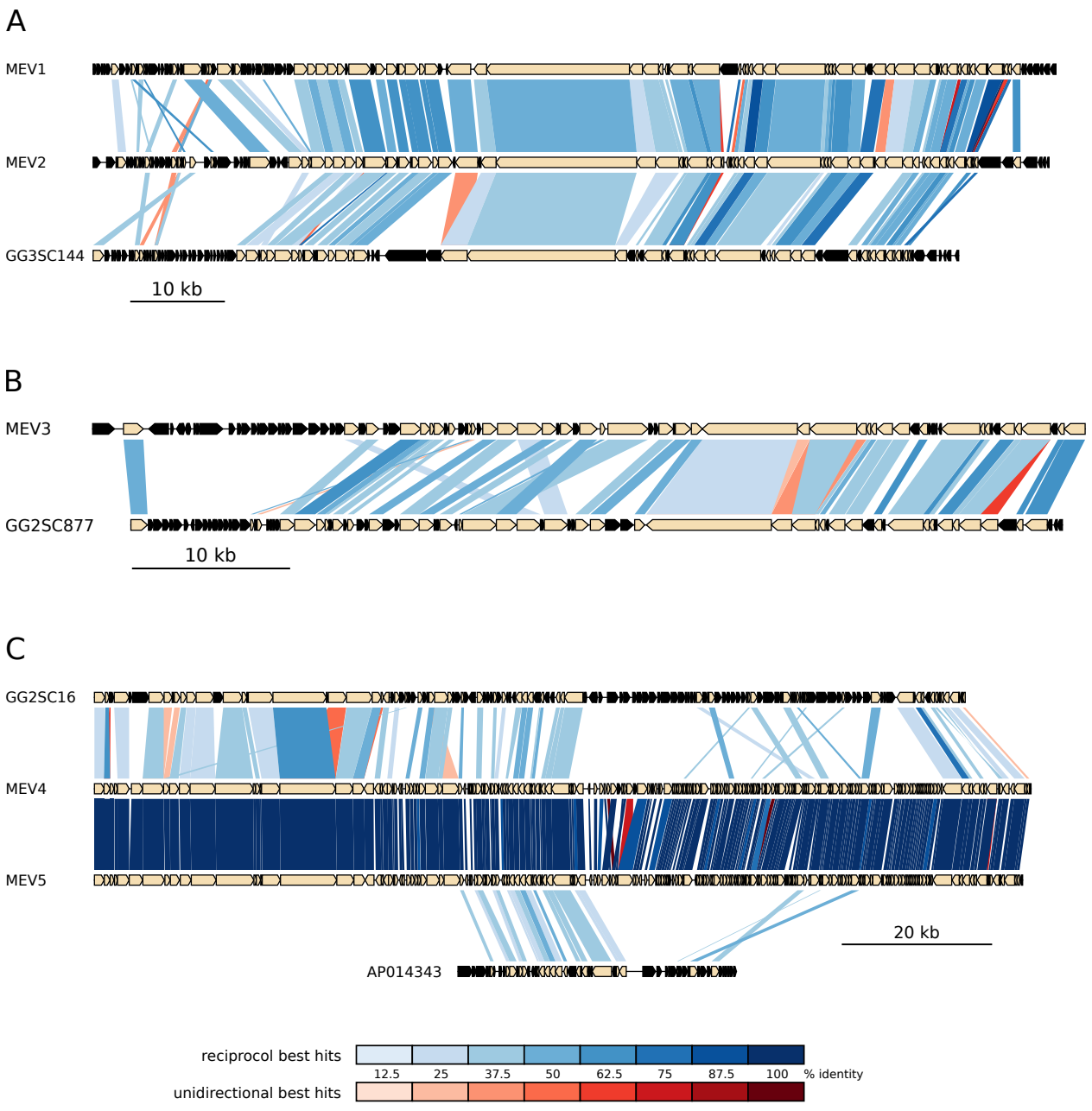

Figure 3.2: Comparison of homologous proteins between genomes and scaffolds of members of the three virus groups. Homologous genes are connected by shaded regions; blue and red indicate reciprocal and unidirectional best hits, and the depth of shading indicates percent amino acid identity on the scale shown. Gene models, depicted as pointed boxes, are shaded black for genes with no blast hits and beige otherwise. Genome and scaffold identifiers are shown for this study, and NCBI accession number shown for environmental fosmid sequence. **(A)** Virus Group 1. **(B)** Virus Group 2. **(C)** Virus Group 3 and fosmid (AP014343).

viruses we examined that remained present in both seasons we sampled. A linear scaffold (GG2SC16) from the October metagenome shares significant protein homology and gene order with about half of the length of the Group 3 genomes. It differs markedly in the genes predicted for the other half of the genome, encompassing a region with dozens of small open reading frames that are conserved between MEV4 and MEV5 but replaced with a different set of short, mostly hypothetical genes in GG2SC16 (Figure 3.2C and Table 3.1). This highly variable region is bounded on the far end by another short region of genes with homology to the Group 3 genomes. Although MEV4 and MEV5 are highly similar, only about 33% of their predicted genes have homologs in scaffold GG2SC16, largely due the variable region just described. Only about 10% of the Group 3 genes were assigned non-hypothetical annotations (Figure A.23 and Table B.18). In addition to the above Group 3 genomes and scaffold, a 40 Kbp environmental fosmid sequence (AP014343) from the Mediterranean Sea was also identified based on conserved order of proteins with homology to MEV4 and MEV5. It represents a likely Group 3 genome fragment, sequenced from a different environment in a distant location [96] (Figure 3.2C).

The sizes of the Group 1 and Group 2 genomes are similar to published Halovirus genomes HVTV-1 (102 Kbp) and HSTV-2 (68 Kbp), respectively, and their genes are similarly organized into two distinct clusters by direction of transcription [112] (Figures 3.2A and A.21). HVTV-1 and HSTV-2 possess linear genomes with terminal repeats at each end [112]. The genomes of groups 1 and 2 may also be linear, having been circularized in our analysis due to terminal repeats at each end collapsing into a single assembled sequence that recruits mate-paired reads from each end of the genome, forming an artifactual loop. This possibility needs to be verified through a follow-up analysis, but the organization of genes on linear scaffolds GG3_SC144 and GG2SC16 is consistent with this interpretation (Figure 3.2A and 3.2B). In contrast to groups 1 and 2, the larger Group 3 genomes have a more complex or-

Table 3.2: Homologous proteins with non-hypothetical annotations shared among the three virus groups. Based on pairwise reciprocal BLAST hits found with an e-value cutoff of 10^{-5}. MG-II is the marine group II *Euryarchaeote* genome (CM001443). Italicized products denote core archaeal genes. Symbols: "X" – Found in all genomes and scaffolds in group; "%" – Found in all genomes, but not additional scaffold; "#" – Found in some genomes and scaffold; "&" – Found only in additional scaffold. Table B.15 is a full version with hypotheical proteins and locus information for each genome and scaffold.

Product	Group 1	Group 2	Group 3	MG-II
thermosome or *GroEL chaperonin	X	X*	X	X
terminase, large subunit	X	X		
phage / plasmid-like protein	#	%		
DNA methylase	#	X		
replication factor C small subunit	X	X		X
DNA polymerase elongation subunit	X	X		X
putative exodeoxyribonuclease	X	X		
putative DNA repair nuclease	X	X		
putative phage tail fiber protein	X	X		
putative P-loop NTPase	X		X	
phage / plasmid-like protein	%		&	
DNA polymerase sliding clamp subunit		%	X	X
DNA methylase		X	X	X
pentapeptide repeats family protein			%	X
DNA-directed RNA polymerase, subunit A			%	X
winged helix-turn-helix DNA-bind protein			%	X
putative hemolysin protein or related			%	X
winged helix-turn-helix DNA-bind protein			&	X

ganization of transcriptional gene clusters, with six clusters on alternating strands ranging in size from ~6–48 Kbp, with no evidence suggesting it has a linear topology (Figures 3.2A and A.23).

Like the HVTV-1 and HSTV-2 genomes, groups 1 and 2 both possess genes consistent with head-tail morphotypes [112], including genes coding putative tail fiber and phage portal proteins. Likely prohead protease, tail tape measure and capsid coding genes are predicted among some of these sequences as well. Genomes of both groups 1 and 2 encode a viral terminase, and have key DNA processing genes including putative methylase, repair nuclease, replication factor C and DNA polymerase genes (Tables 3.2 and B.15) [11]. The DNA polymerases in both groups cluster phylogenetically with those found in Haloviruses and their hosts among the euryarchaeal *Halobacteria* (Figure A.24). This evidence, combined with preservation of relative gene order between the genomes of groups 1 and 2 suggests these are head-tail viruses with linear genomes, distantly related to one another and to well characterized lytic viruses infecting halophilic *Euryarchaeota* (Figure 3.3) [112].

The Group 3 genomes encode a small number of proteins homologous with one of the other two groups, most notably a subunit of the archaeal DNA sliding clamp complex [57, 11, 20] shared with Group 1, which phylogenetically clusters with the homologous protein found in the marine group II *Euryarchaeote* genome (Tables 3.2, B.15 and Figure A.25). The Group 3 genomes encode a putative phage integrase, RNA polymerase, transcription initiation factor, and glycosyltransferase proteins that could point to a lysogenic reproductive cycle [161, 165]. Suggestively we identified a metagenomic scaffold (`GG2SC40`) in the October sample that contains two regions of different coverage and GC content corresponding, respectively, to a region of the marine group II genome, and a fragment of a potentially integrated provirus genome related to the Group 3 virus genomes (Figure A.28). If confirmed this would be significant, as all currently characterized euryarchaeal viruses reproduce via

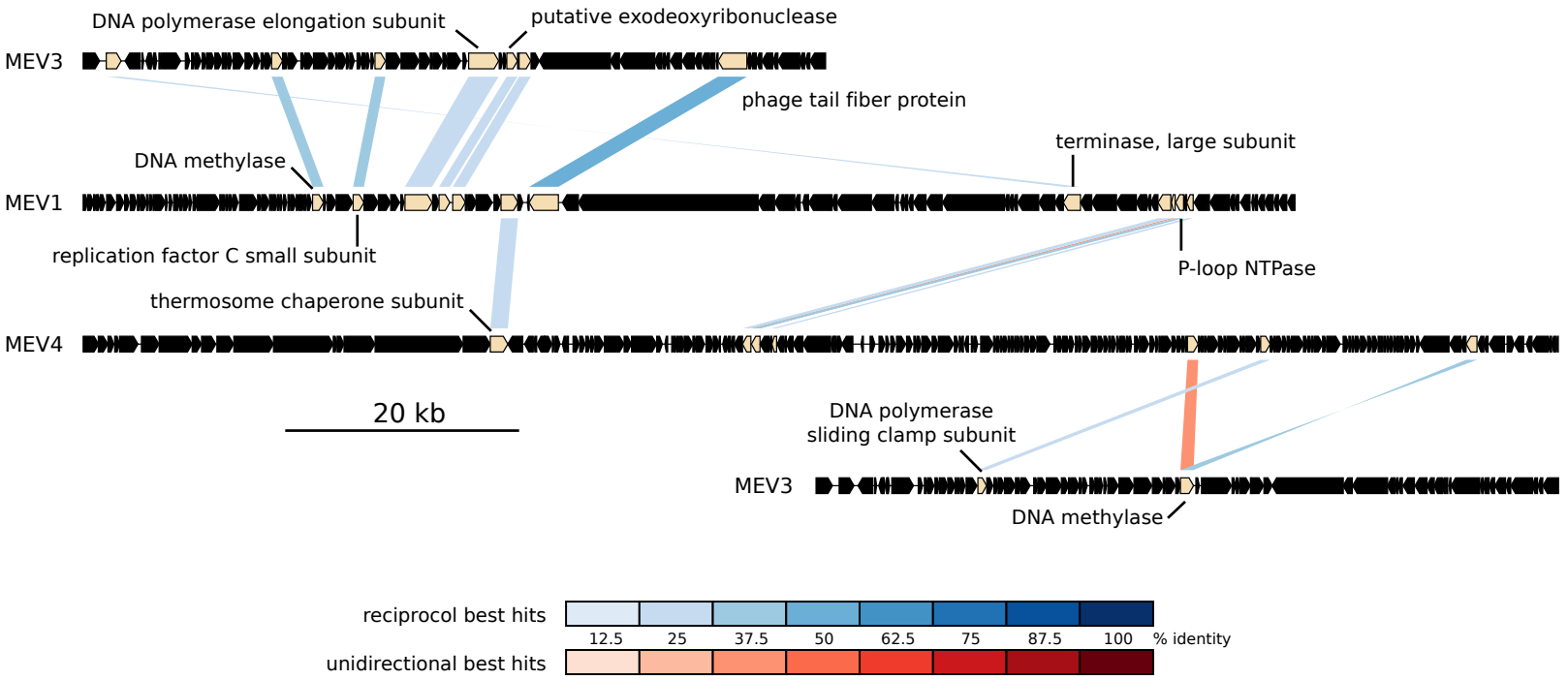

Figure 3.3: Comparison of homologous proteins between selected members of each of the three virus groups (Group 1: MEV1, Group 2: MEV3, Group 3: MEV4, repeated at top and bottom). Homologous genes are connected by shaded regions; blue and red indicate reciprocal and unidirectional best hits, and the depth of shading indicates percent amino acid identity on the scale shown. Gene models, depicted as pointed boxes, are shaded black for genes with no blast hits and beige otherwise. Genes with non-hypothetical annotations are noted.

lysis of the host cell, and only a single putative provirus has been identified among all archaeal genome sequences [75, 138].

3.2 Viral chaperonins

Genes coding for HSP60 family [78] protein chaperonin subunits are present in the genomes from all three groups of MEV viruses, including the additional metagenomic scaffolds and the putative integrated prophage sequence on scaffold `GG2SC40` (Tables 3.2 and B.15, Figure A.28). Presumably all three groups of viruses have proteins that require special assistance to reach their native folded state within the host cell, and carry these genes to augment the host chaperonin complexes, as has been noted for a few bacteriophages [76, 25].

The best studied HSP60 chaperonins are the nearly universal bacterial groEL/ES complexes, which are homo-oligomers of groEL proteins with 7-fold symmetry, associated with an equal number of groES co-chaperonins [86] (Figure 3.4A). Archaea also have an essential HSP60 chaperonin complex known as the thermosome [110, 62]. In most archaea the thermosome is a hetro-oligomer with 8-fold symmetry, made up of two alternating paralogous (α- and β-subunit) proteins [103, 81]. Other Archaea generate homo-oligomeric 8-fold symmetric thermosomes from a single subunit [5], whereas a few Archaea make thermosomes with 9-fold symmetry from three alternating paralogous (α-, β- and γ-subunit) proteins [72, 90, 6] (Figure 3.4B-C). The MG-II genome encode three thermosome paralogs, although it is unknown if they are arranged in hetro-oligomers with 9-fold symmetry (Figure 3.4B-C).

Unexpectedly, the predicted chaperonin proteins encoded by the Group 2 virus sequences cluster phylogenetically with bacterial groEL proteins (Figures 3.5 and A.29), yet Group 2 genomes do not appear to encode a gene for the essential groES co-chaperonin protein. The MG-II genome and known marine group II environmental sequences lack groEL or groES

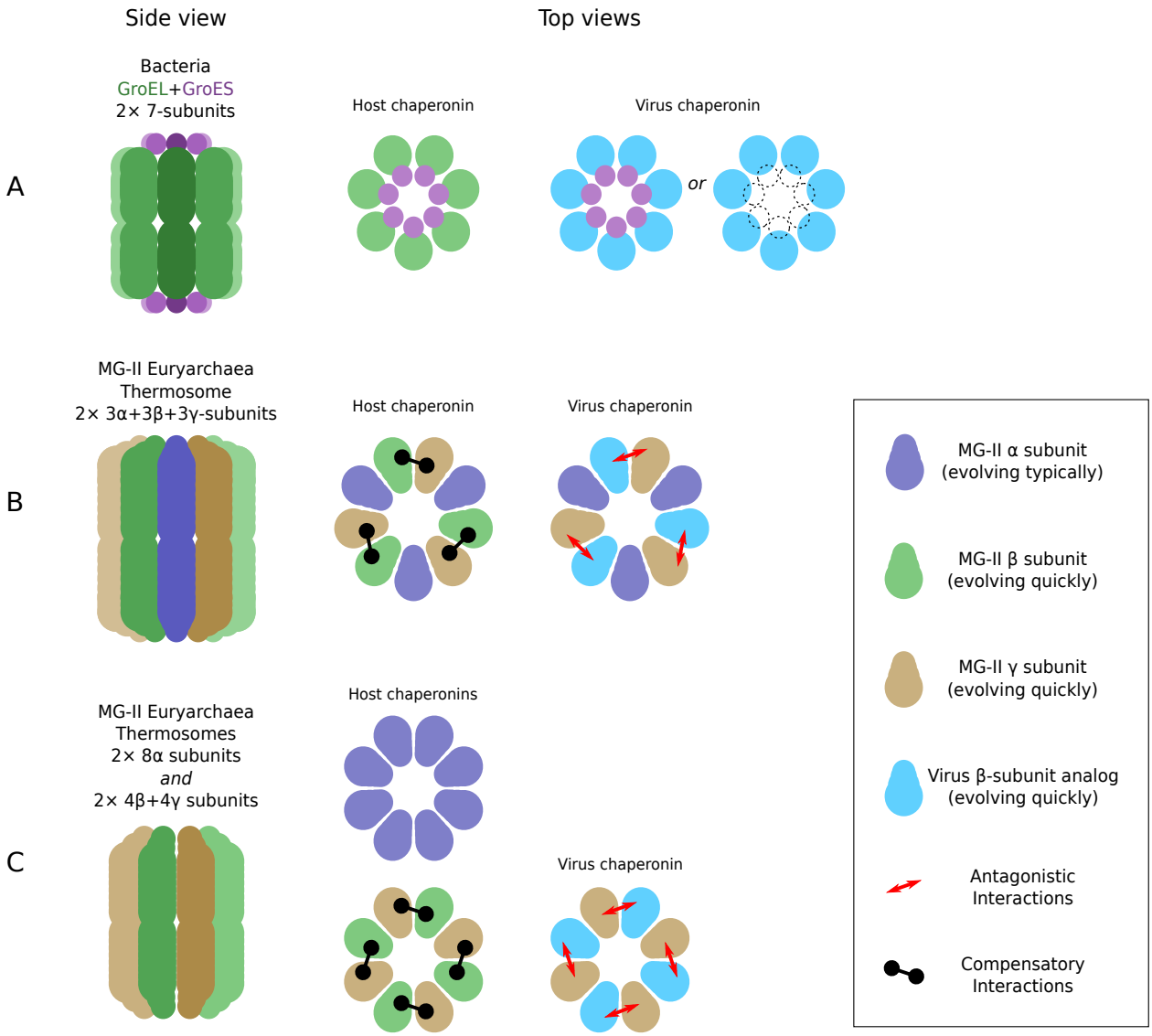

Figure 3.4: Putative viral chaperonin subunit interactions with with host proteins. **(A)** Typical groEL/GroES co-chaperonin complex with 7-way symmetry. Marine group III *Euryarchaeota* have groEL/groES, but Group 2 viruses only have groEL. Viruses may use the host groES or may forgo use of a co-chaperonin. **(B)** Marine group II *Euryarchaeota* have α, β, and γ thermosome subunits and Group 1 and 3 viruses also have a subunit. *Sulfolobus shibatae* forms α,β,γ-hetero-oligimer complexes with 9-way symmetry. Rapidly evolving virus and host β- and γ-gamma subunits may be explained for this case by the depicted interactions. **(C)** Two subunit hetero-oligimer complexes with 8-way symmetry are common among Archaea. 8-way homo-oligimer complexes also occur. Rapidly evolving virus and host β- and γ-gamma subunits may be explained for this case by the depicted interactions.

homologs, although the presence of groEL/ES has been reported in a methaogenic *Euryarchaeote* [71]. We uncovered evidence of groEL/ES in fosmid sequences retrieved from Genbank that can be attributed to marine group III *Euryarchaeota* using ribosomal protein phylogeny [36] (Figures 3.5 and A.29). It is likely that the Group 2 virus groEL subunit requires a groES co-chaperonin to function, so its hosts are most likely marine group III (or unknown marine group II) *Euryarchaeota* that possess groEL/ES chaperonin genes. No marine group III *Euryarchaeota* were detected in the October or May metagenomes [64], and Group 2 virus sequences were only assembled from the October metagenome, so the identity of the Puget Sound marine *Euryarchaeote* host of these viruses is a open question.

Group 1 and 3 viruses encode chaperonin subunits that cluster phylogenetically with archaeal thermosome proteins (Figures 3.5 and A.29). Two of the three thermosome paralogs (β- and γ-) present in the MG-II genome are significantly diverged from other Archaea, as well as the third (α-) MG-II paralog, with only 23-32% amino acid identity between aligned sequences. By contrast, aligned α-, β- and γ-subunits from genomes representing three genera of euryarchaeal *Halobacteria* have pairwise amino acid identities ranging from 42-84% among the nine proteins (data not shown).

To understand the scope and potential evolutionary underpinnings of these diverging host and viral paralogs, we identified seven additional marine group II thermosome coding sequences in our Puget Sound metagenome assembly scaffolds (Table B.19, Figures A.27 and A.28) and 27 additional sequences from euryarchaeal environmental fosmids [36] in Genbank that cluster phylogenetically with the marine group II α-, β- or γ-subunits (Figures 3.5 and A.29). Within the resulting β- and γ-subunit clades we found pairwise aligned amino acid identities as low as 31% and 32% respectively. Whereas within the α-subunit clade, the minimum amino acid identity was 70%. We identified an additional clade of marine euryarchaeal thermosome sequences from environmental fosmids that were linked by

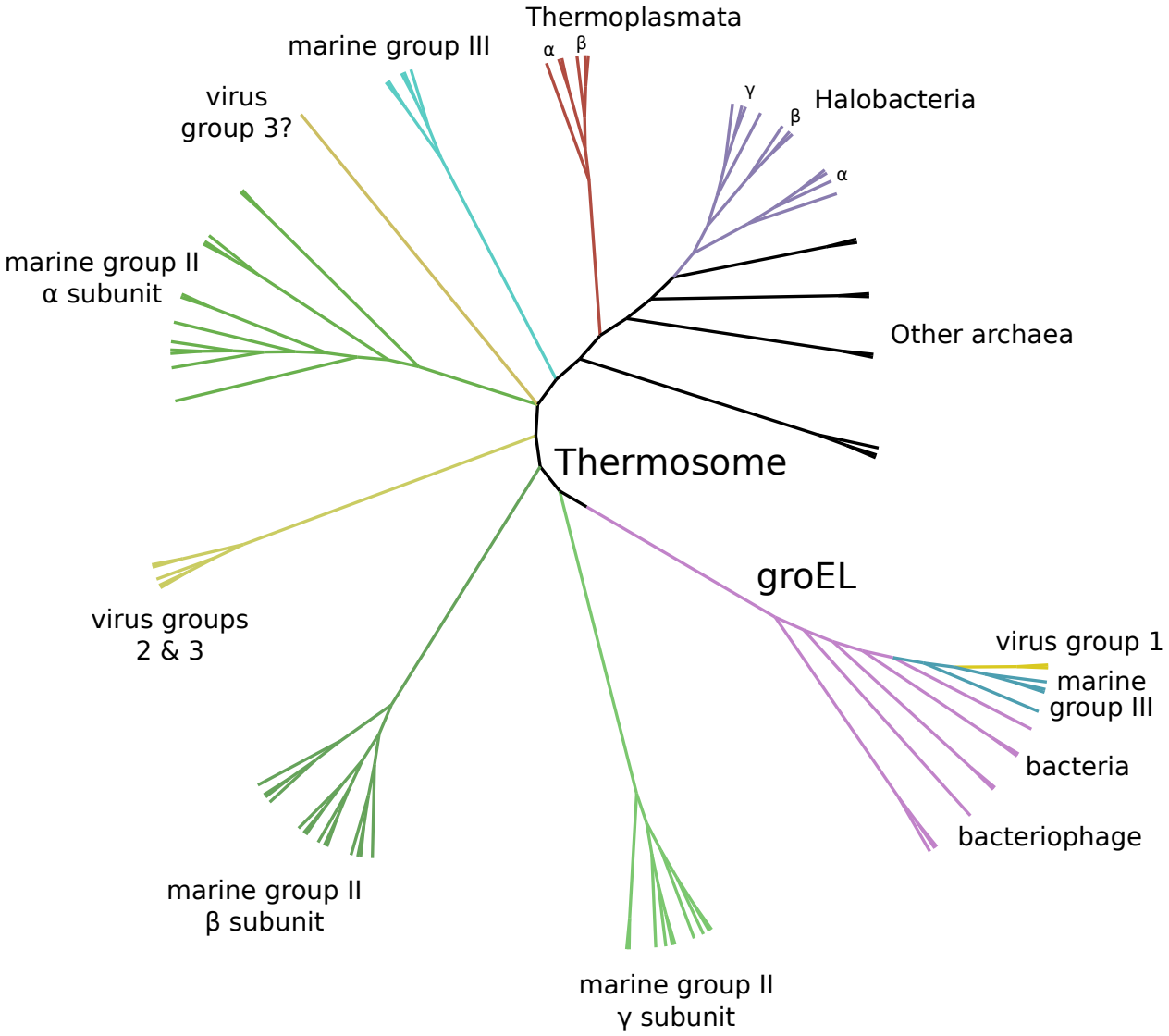

Figure 3.5: Unrooted cladogram depicting Bayesian phylogeny of selected thermosome (HSP60) chaperonin proteins. Major clades are labeled taxonomically, with α, β and γ denoting paralogous subunit proteins. groEL chaperonin proteins from selected Bacteria, Archaea and bacterial and archaea viruses are also shown. Branches containing virus proteins from this study are colored in yellow. See Figure A.29 for the corresponding detailed phylogenetic tree.

ribosomal protein phylogeny to the marine group III *Euryarchaeota* (Figure A.26), and found amino acid identities between these marine group III thermosome proteins and those of the marine group II α-paralogs to be ~50%, dramatically higher than among the α-, β- and γ-paralogs within the MG-II genome itself (23-32%, Figures 3.5 and A.29).

These data show that the thermosome proteins of the Group 1 and 3 genomes and those of globally distributed marine group II β- and γ-paralogs are all evolving at a remarkably rapid rate relative to the divergence seen between members of different classes of *Euryarchaeota*; while the marine group II α-subunit is evolving at a rate much more typical of that seen in other groups of *Euryarchaeota*. The simplest explanation is that these viruses require chaparonins with modified subunits to correctly recognize and fold one or more of their own proteins, and the Group 1 and 3 virus genomes encode analogs that can substitute for the host β- or γ-subunit paralog to produce a mixed virus/host hetero-oligimer complex with 8 or 9-fold symmetry. (Figure 3.4B-C). We interpret 9-fold symmetry to be the most parsimonious for marine group II *Euryarchaeote* thermosomes, given the slower evolving α-subunit and lack of reports of Archaea with three thermosome paralogs constructing 8-fold symmetric homo-oligimeric thermosome complexes (Figure 3.4C).

In either arrangement, we hypothesize that the non α-subunit host proteins are under intense selective pressure to antagonize the viral analog by physically excluding it from the complex or by inhibiting the ability of complexes with the viral analog to recognize and properly fold viral proteins into their native conformation. The host subunit substituted by the viral analog is under selective pressure to compensate, rapidly adapting to changes in the antagonistic paralog to maintain the functionality of the host thermosome complex. The struggle between these viruses and their hosts for control of the thermosome represents a clear molecular example of a classic evolutionary arms race between predator and prey [28, 18, 19, 53, 35].

3.3 Concluding remarks

Mesophilic marine *Euryarchaeota* are hosts for at least three distinct groups of dsDNA viruses, two of which appear to be head-tail viruses with some similarity to known viruses infecting extremophic euryarchaeal *Halobacteria*; while a third group may represent the first known euryarchaeal lysogeneic viruses. Notably, all three virus groups carry protein chaperonin genes—the first archaeal viruses identified to do so—revealing an global and apparently ancient evolutionary arms race between viruses and the marine group II *Euryarchaeota* for control of essential host protein folding machinery.

Chapter 4

METHODS

4.1 Sample collection, preparation and sequencing

Samples for metagenomic analysis were collected on October 10, 2008 (October) and on May 4, 2009 (May). Approximately ninety liters of surface water (1m) were collected from the Puget Sound main basin in Seattle, WA (47° 41.24' N, 122° 24.14' W). The less than 0.8μm fraction was concentrated by tangential flow filtration (TFF) equipped with a 30kD Biomax Pellican 2 Cassette (Millipore Corporation, Billerica, MA). Cells were concentrated to a final volume of approximately 150mL. Cells were subsequently pelleted by centrifugation at 4°C for 60 min at 17,000×g and resuspended in 20mM Tris (pH 7.4). DNA was extracted from the concentrated cells using a DNeasy Blood and Tissue kit (QIAGEN, Germantown, MD) following the manufacturer's instructions.

4.1.1 Library Construction and SOLiD Sequencing

Metagenomic libraries were constructed according to the SOLiD™ v3.0 mate-pair library preparation protocol (Life Technologies, Foster City, CA). Briefly, 22.2 μg (October) and 20.6 μg (May) of input DNA were sheared and sized selected to produce DNA fragments between 2000 and 3000 base pairs. The DNA was end repaired, ligated with EcoP15i CAP adaptors and circularized. Subsequently, the circularized DNA was nick translated and digested, leaving 50 bases of genomic DNA on either side of the internal adaptor. Finally P1 and P2 adaptors were ligated to the ends of the genomic DNA and served as sites for PCR amplification. Lambda DNA was also prepared for mate pair sequencing, was added

to the metagenomic library at a ratio of 1:1000 (Lambda: Genomic DNA) and served as an internal control. Libraries were deposited on full slides and sequenced using the SOLiD™ v3.0 system. The October library was sequenced by Life Technologies (Foster City, CA) and the May library was sequenced at the University of Washington, Center for Environmental Genomics (Seattle, WA).

4.1.2 Cell Counts

Cell counts and TFF recovery efficiency were determined by staining cells from whole water and concentrated seawater samples with the nucleic acid stain 4',6- diamidino-2-phenylindole (DAPI) and by fluorescence *in situ* hybridization (FISH) as previously described [100]. Cells were imaged using a Nikon 80i microscope equipped with a CoolSNAP HQ2 camera (Photometrics, Tucson, AZ) and NIS Elements Basic Research software (Nikon Instruments, Melville, NY). More than 500 cells were counted from each slide (15 frames). Count data is summarized in Table B.1.

4.1.3 Clone library construction

For the October sample bacterial and archaeal 16S rRNA gene clone libraries were constructed. Briefly, RNA genes were amplified from community genomic DNA by PCR using Taq polymerase (Genechoice, Frederick, MD) and variations of commonly used bacterial primers, 8F and 1492R, or archaeal primers, 21F and 1492R. Amplifications were performed in a C1000 thermal cycler (Bio-Rad Laboratories, Hercules, CA) using the following conditions: 35 cycles, annealing at 55°C for 1 min, elongation at 72°C for 2 min, and denaturation at 94°C for 30 sec. A single band of the predicted length was observed by agarose gel electrophoresis, excised and purified using a Minelute Gel Extraction kit (QIAGEN, Germantown, MD) according to the manufacturer's instructions.

Clone libraries were constructed and sequence identities were obtained as described by Morris et al. [99]. Briefly, amplicons were cloned into the pGEM T-easy vector (Promega, Madison, WI) and sequenced at the University of Washington High-Throughput Genomics Unit (Seattle, WA). Cloned 16S rRNA gene sequences were aligned to a custom ARB database [87] that contained 151,952 sequences from cultured organisms and environmental gene clone libraries. Taxonomic assignments were determined by phylogenetic inference.

4.2 Read preparation, alignment and classification

4.2.1 Metagenome read preparation for alignment and assembly

Raw color-space SOLiD reads were converted to the FASTQ format for compatibility with standard tools, de-duplicated to remove redundant PCR duplicated mate-pairs, and trimmed/filtered for quality, all using custom software included with the SEAStAR tools (see Appendix 3). Conversion to FASTQ was accomplished with the `solid2fastq` tool. PCR duplicate mate-paired reads were removed using the `fastq_nodup` tool, which implements a method that indexes and compares mate-paired reads directly in an error tolerant manner, removing the lowest quality pairs when duplication is detected. For trimming we used the `trimfastq` tool, and quality scores were used to determine the cumulative probability of zero uncorrected nucleotide-space errors occurring up to each position in a read. Reads used for alignment from each sequencing run were independently trimmed using an error probability threshold tuned to maximize the sequence aligned with a known reference genome. For the October sample, the genome sequence of *Pelagibacter ubique* str. HTCC1062 (`NC_007205`) was used, yielding a trimming threshold of 0.4. For the May sample, the genome sequence of *Enterobacteria* phage lambda (`NC_001416`, an added standard) was used, yielding a threshold of 0.5. These values correspond with the predicted probability that

a trimmed read is free of uncorrectable nucleotide errors, based on instrument generated quality scores. Reads trimmed below 34nt in length were discarded, as were low complexity reads with measured mean entropy of less than 3 bits per di-nucleotide after trimming. Reads containing low complexity sequence were discarded because this is a common error mode for SOLiD generated sequence, and such reads consume memory and complicate *de novo* assembly.

For *de novo* assembly, a further reduced set of higher quality reads was produced using the above methods, but tuning thresholds were determined to produce the approximate number of reads that would fit in 144 Gbytes of available memory during the assembly process (error thresholds 0.76 and 0.93 for the October and May samples, respectively). Quality trimmed reads used for assembly were then error-corrected using the Applied Biosystems SOLiD™ Accuracy Enhancer Tool (SAET) tool version 2.2 (Foster City, CA) with an expected reference length of 200 megabases and parameters `-trustprefix 25 -localrounds 3 -globalrounds 2`. These settings were selected as a reasonable tradeoff between error correction potential, and memory use and computation time required by the tool.

4.2.2 Alignment to reference databases

Quality trimmed reads from each sample were aligned to reference databases using BWA [83] version 0.5.7. 16S rDNA alignment was performed against the "good quality, ≥ 1200bp" subset of the Ribosomal Database Project (RDP) [26] database release 10 update 25, with the omission of sequences with significant homo-polymer runs, and the addition of clone library sequences from the October sample. Whole genome alignment was performed against the NCBI RefSeq [117] database release 46, with the omission of Eukaryotic nuclear genomes. Metagenome alignment was performed against as-

sembled contigs from the Global Ocean Sampling [130] "GOS: Assembled Sequences (N) – Scaffolds and Unassembled Sequences" database, downloaded from CAMERA as file `node1015439915564925280.fasta.gz` on October 10, 2009 [133]. All BWA alignments were produced using the following parameters:

```
bwa aln -c -n 0.001 -l 18 -k 2
bwa samse -n 500000
```

These parameters were selected to allow a tradeoff between allowing a liberal number of mismatches between reads and reference sequences (varying depending on trimmed read length) and maintaining reasonable computational complexity given the multi-gigabase reference databases used. Significantly, these settings also explicitly return multiple candidate alignments (up to 500000) for any read that aligns equivalently well with more than one position in a reference database.

4.2.3 Alignment post-processing for reference selection and abundance/coverage estimation

Read alignments in the SAM format [84] output by BWA were post-processed using the SEAStAR `ref_select` tool. Briefly, `ref_select` works by calculating an information bit-score [134] for each aligned read by treating it as an encoded symbol and using the number of aligned positions in the reference sequence database to estimate its probability. The information content I (as a bit-score) of each read aligned to one or more reference sequences in a given database was quantified as: $I(p) = -\log_2(p)$ where p is the measured probability of the read aligning with a randomly selected reference sequence in that database [134]. The most parsimonious database reference sequences (those that best explain the information contained in the read-reference alignments) are selected using a greedy algorithm that iteratively chooses sequences with the maximum length-normalized residual total aligned read bit-score, down to a floor threshold. The residual bit-score for a reference is the sum

of the bit-scores of aligned reads that do not also align with previously selected reference sequences. This algorithm produces the set of "selected reference sequences" used by the following analyses.

Coverage is estimated by processing reads in order of bit-score (high to low), allocating coverage to selected reference sequences fractionally among each read's alignments, weighted by the relative mean coverage already accumulated among those reference sequences. That is, coverage is accumulated successively, with priority given to the most informative reads; those that align uniquely and those that align to the fewest positions among selected reference sequences. In this way, the coverage contributed by reads that map to multiple selected reference sequences is shared fractionally among those references, with the split proportion determined by more informative reads. Fractional abundance is estimated for each selected reference sequence as its fraction of the sum of the mean coverage of all selected sequences (see Figure A.1 for an illustrated example of this process).

Note, this method allows reference sequences that are selected but not fully covered by metagenome reads to be identified; indicating that novel taxa (not found in the database) are present in the sample. If desired, the covered (conserved) portions of the selected reference sequence can be used to design primers facilitating targeted reconstruction of the novel taxa using Sanger sequencing. We did not use this approach because our October 16S rDNA clone libraries recovered all taxa with estimated abundance greater than 1%.

4.2.4 Bayesian classification of environmental 16S rDNA sequences

To assign taxonomic information to full-length RDP and October clone library 16S rDNA reference sequences selected by `ref_select` (Figures A.5 and A.8), these sequences were automatically classified using the RDP Bayesian Classifier [159] version 2.2 using a custom training set comprised of the standard RDP training set (version 6_032010) with the

addition of taxonomy and Genbank training sequences for major environmental clades detected in our October clone libraries (resulting from ARB alignments previously described) that the standard RDP training set did not encompass (Figure A.3). A previously unnamed novel clade of marine alpha-*Proteobacteria* that was detected in the October clone libraries and both of our metagenomic samples has been dubbed the "PS1" group. Figure A.4 shows a bootstrapped neighbor-joining phylogeny including this group, computed from the standard RDP [26] alignment using the software package Geneious Pro [37] version 5.3. All October 16S rDNA clone library sequences and the reference sequences selected by `ref_select` from the October and May metagenome alignments with the modified RDP database were classified based on the custom training set described above (results in Figures A.2, A.5, and A.8, respectively).

4.2.5 SEAStAR simulations and 16S database comparison

The specificity and accuracy of SEAStAR `ref_select` reference sequence selection and abundance estimation were evaluated through simulations using synthetic metagenomic datasets generated from randomized populations of 16S rDNA sequences. 10 random trial populations of 28 family-level taxa were drawn randomly from the October 16S rDNA clone library sequence classifications (excluding sequences that did not classify with $p \geq 0.85$, and taxonomic orphans, as shown in bold italic type on Figure A.2). For each trial population, a clone sequence was randomly selected to represent each selected family-level taxon, and each clone sequence was randomly and exclusively assigned one of four relative abundance classes: 9.8668% (~10.0), 3.1202% (~3.0), 0.9867% (~1.0) and ~0.3120% (~0.3), with each class containing 7 clones, yielding a total abundance of 100%. The four exact relative abundance class values used were selected to be equally spaced on a log scale while summing to 100%.

Each trial population was used to generate two large simulated read sets ($n = 106$ reads) with errors introduced based on models for the October and May SOLiD sequencing runs respectively, using the `simutrain` and `simulate` tools in the Maq [85] software package version 0.7.1, with custom modifications to correctly handle generation of color-space reads. The resulting read sets contained reads generated from each clone sequence in the correct proportion for the trial population's family-abundance class it was selected to represent.

A subset of reads ($n = 80000$, approximately the number of reads from our October and May metagenomes aligning with the RDP database) was randomly drawn from each simulated read set for use as the simulated population sample. Each of the 20 read population samples was separately aligned against three 16S rDNA databases: October clones only, October clones added to the RDP database, and the RDP database only.

Alignments, post-processing using `ref_select`, and taxonomic classifications were performed as described above for the actual metagenome reads, with the exception that the 28 clone sequences used to generate each simulated read set were excluded from all alignment databases used for that read set. The resulting classifications and estimated abundances for each synthetic read set were compared to the corresponding simulated starting population and binned at the family-level as successful detections, false positives and false negatives. Detection and population abundance estimation results are summarized in Figure A.7.

The effect of the sequence composition of 16S rDNA databases was further evaluated by aligning the October metagenome reads to the same three databases used for the simulations described above, run through the same analysis pipeline, and compared with the 16S abundances implied by counting classified clone library sequences (as shown in Figure A.2). The results of this comparison are shown in Figure A.6.

4.3 Metagenomic assembly, scaffolding and binning

4.3.1 De novo contig assembly of metagenomic reads

Color-space contigs were assembled from the SAET error-corrected read sets from each of the two sample metagenomes using the program Velvet [168] version 0.7.63 with a k-mer size of 21 and parameters:

```
-scaffolding no -read_trkg yes -ins_length X -ins_length_sd 210
-exp_cov 1000 -cov_cutoff 4 -min_contig_lgth 75
```

where the insert length `X` was estimated from reference alignments to be 1100 and 2100 for the October and May samples, respectively. The effect of k-mer size was evaluated for all odd values between 16 and 28, with 21 selected because it maximized the N50 contig length metric. All other parameters were selected based on recommendations in the Velvet documentation. The resulting contigs are summarized in Table 2.1.

4.3.2 Re-alignment of reads to contigs, mate-pair processing, visualization and scaffolding

Quality trimmed reads from the each sample metagenome were aligned back to the corresponding color-space assembly contigs using BWA [83] as described for the reference databases. Color-space contigs were naively converted to nucleotide-space using a constant nucleotide prefix for the purposes of input to BWA, which requires nucleotide-space reference sequences but immediately converts them back to color-space internally when the `-c` parameter is used. The resulting SAM alignment files were post-processed with the SEAStAR `ref_select` tool to produce nucleotide-space consensus contigs with coverage estimates and bit-scored mate-pair connections between contigs.

Briefly, `ref_select` uses a custom dynamic programming algorithm to generate consensus nucleotide-space contigs from color-space read alignments, and assigns bit-scores to

mate-pair connections within and between contig sequences based on the sum of the minimum alignment bit-score (calculated as described above) assigned to each pair of mated reads. `ref_select` emits a mate-pair connection graph which represents contigs as nodes, mate-pair connections as edges, and encodes for each element of the graph, statistics such as total bit-score, coverage, %GC content, sequence length, and mean aligned positions of mate-paired reads. These output graphs were converted to DOT format using the SEAStAR `graph_ops` tool, post-processed (filtered by bit-score and colored by %GC) and visualized using the `neato` tool from GraphViz [46] version 2.27 (Figures A.9 and A.10).

Sequence scaffolds were produced using the SEAStAR `graph_ops` tool for each sample through a series of operations on the mate-pair connection graphs described above. First, all low-quality connections were filtered out of each graph by removing edges with a mate-pair bit-score below a minimum threshold of 25 bits. Each filtered graph was transformed into a set of maximum spanning trees with edge weights determined using a heuristic taking into account mate-pair bit-scores and the estimated coverage and %GC content of the connected contigs. In this way, the remaining connections maximized parsimony by eliminating the weakest connections and those between contigs with divergent %GC and coverage estimates.

Branches containing contigs totaling more than 5000bp of sequence (the minimum length anticipated to have usable tri-nucleotide usage statistics, see below) were cleaved off the tree by selectively removing the weakest mate-pair edges, determined using the same weighting heuristic used to calculate the maximum spanning tree. The remaining connected components were laid out into scaffold sequences (with gaps) using the SEAStAR `seq_scaffold` tool, which automatically determines the order and orientation of contigs from mate-pair alignment statistics. Potential chimeric (misassembled) contigs were detected automatically using the `graph_ops` tool, which searches for anomalous drops in mate-pair

physical coverage within contigs (i.e. few mate-pairs spanning a particular region). These cases were individually curated by hand, with contigs split at such positions when necessary.

4.3.3 Binning scaffolds by genome

For each sample metagenome, the SEAStAR tool `tetracalc` was used to bin assembled scaffolds into candidate genomes based on aligned read coverage and a statistical analysis of nucleotide usage patterns. Briefly, tri- and tetra-nucleotide usage anomaly Z-statistics [146] were calculated for each scaffold and linear regressions of the resulting 256 (or 64, for tri-nucleotides) values were calculated for each pair of scaffolds (e.g. Figure A.12).

Scaffolds were clustered by building a graph of all scaffolds (nodes) connected by edges representing tetra-nucleotide Z-statistic correlation coefficients exceeding an empirically determined threshold ($R > 0.9$). Additional edges were then added for lower tetra-nucleotide correlations exceeding a lower threshold ($R > 0.7$) when at least one of the scaffolds in a pair contained less than 30Kbp of sequence and the tri-nucleotide correlation exceeded a third threshold ($R > 0.85$). Connected components of each resulting graph containing more than 950Kbp of scaffold sequence became the candidate genome bins for each metagenome sample.

Candidate genome bins were then phylogenetically screened using aligned mate-pair connections between informative regions of 16S rDNA sequences selected from RDP database alignments (as described above), and positions within the candidate genomes. The set of reads mapping uniquely to selected 16S rDNA sequences classified to the same genus by the analysis of RDP database metagenome alignments were identified as "16S taxonomic anchors" for each sample. Contig alignments for mate-pairs of these 16S anchor reads were identified, and taxonomic assignments were made for scaffold bins containing contigs connected (via mate-pairs in anchors) to 16S rDNA sequences classifying to a single genus.

*4.3.4 Alignment of binned scaffolds to the alpha-*Proteobacterium *str. HTCC2255 genome*

The candidate genome bins from the October sample included a set of scaffolds with contigs connected to a 16S sequence (RDP ID: `S000380128`) classified as genus *Thalassobacter*. This sequence corresponds identically to the 16S sequence from the genome sequence of alpha-*Proteobacterium* str. HTCC2255 (`NZ_DS022282`), which was also identified as the most highly covered genome from the analysis of the October sample alignment with the RefSeq database (Tables B.2 and B.3). This candidate genome bin contained 41 scaffolds (grey highlighted sequences, Figure A.9) which were then aligned in nucleotide-space with the alpha-*Proteobacterium* str. HTCC2255 reference genome using the `nucmer` tool from the MUMMER [77] software package version 3.22, using alignment parameters `-b 1500 -g 500 -c 500` and plotted using the `mummerplot` command with option `--layout` (Figure A.11).

4.4 Finishing, verification and annotation of the MG-II genome

4.4.1 Assembly of binned scaffolds into the marine group II Euryarchaeote *genome*

The screened candidate genome bins from the May sample included a bin containing a set of 16 scaffolds (grey highlighted sequences, Figures 2.2 and A.10) with contigs connected to the informative regions of 16S sequences classified as marine group II *Euryarchaeota* (Figure A.6). This candidate genome was selected for full assembly.

All quality filtered metagenome reads from the May sample were aligned to contigs in the marine group II *Euryarchaeote* (MG-II) scaffolds, and the results analyzed with the SEAStAR `ref_select` and `graph_ops` tools, yielding a mate-pair graph forming a single connected component. The set of quality filtered reads recruited in the alignment and their mates were then reassembled using Velvet [168] version 1.0.13 with a k-mer size of 21 and parameters:

```
-scaffolding -no_read_trkg -no_ins_length -auto_ins_length_sd
-auto_exp_cov 1000 -cov_cutoff 13 -min_contig_lgth 75
```

This reassembly process was repeated nine times, until the N50 of the assembled contigs stabilized. The final assembly of 743 contigs (N50 = 6819) was processed by aligning all quality filtered metagenome reads from the May sample back to the contigs, analyzing the SAM output with `ref_select` and processing the resulting output graph as described above for the production of metagenome scaffolds. The resulting 11 scaffolds were hand checked for chimeric contigs and layout problems. In this process a highly repetitive region was identified, and a best effort was made to lay out its contigs by hand into the "Repeat region" (RPR). The curated assembly of 11 scaffolds plus the repeat region formed the "draft assembly" of the MG-II genome.

The final assembled version of the MG-II genome was hand curated. Many contigs were merged by identifying unassembled overlaps between neighboring contigs within a scaffold using the Geneious Pro [37] assembler version 5.3. A candidate circular arrangement of the scaffolds was determined using the highest scoring mate-pair connections at the scaffold boundaries, and several ambiguous cases were resolved using PCR and Sanger sequencing (see next Method section). This process yielded a single linear scaffold layout with "dangling" mate-pairs connecting to relatively low coverage metagenome contigs at each end.

A process of iterative reassembly similar to that described for the whole genome above, but with relaxed minimum coverages: `-cov_cutoff 1` (first 20 iterations) and `-cov_cutoff auto` (subsequent iterations) was then employed. Proceeding from the last contig on each end of the scaffold layout, after 40 iterations a series of short mate-pair connected contigs forming two distinct low coverage paths across this final gap were observed; each estimated from the mean mate-pair insert size to be less than 20Kbp long.

One of these paths was selected, and a complete assembly across the gap along that set of connected contigs was produced using Sanger sequencing (see next Method section) and the Geneious Pro [37] assembler version 5.3, and subsequently verified by realigning with

May metagenome reads, obtaining complete coverage across the gap. This assembled sequence forms the ~14 Kbp Hyper-Variable Region (HVR) of the Mar. Group-II final assembly. Quality filtered metagenomic reads from the May sample were aligned back to the final MG-II assembly, the results analyzed with `ref_select`, and position specific genome statistics were generated (Figure A.13).

4.4.2 PCR amplification and Sanger sequencing across MG-II assembly scaffold gaps.

Gaps between SOLiD Sequencing scaffolds were closed by designing custom primers to span the gap. Amplification was performed in a C1000 thermal cycler (Bio-Rad Laboratories, Hercules, CA) using varied temperatures and extension times based on the melting temperatures of the primer sets and the anticipated length of the product. Amplification products were visualized by gel electrophoresis and purified from the gel using the Qiagen Minelute Gel Extraction Kit (Qiagen, Gaithersburg, MD). The resulting purified DNA was cloned into a TOPO TA Cloning Kit for Sequencing vector (Invitrogen, Carlsbad, CA) according to the manufacturers instructions. Plasmids containing the insert of interest were purified using the Qiaprep Spin Miniprep Kit (Qiagen). The resulting plasmids were sequenced at the University of Washington Department of Biochemistry using a ABI 3730XL sequencer (Foster City, CA).

4.4.3 Genome annotation

A preliminary annotation of the draft MG-II genome was prepared using the RAST [9] service version 4.0. This annotation was used to hand adjust the length of a few small gaps (denoted with 15-17 Ns) in the assembly to avoid the introduction of artifactual frame-shifts within genes that span these gaps. Noncoding RNA genes were identified using RNAmmer [80] version 1.2 (Figure A.14). Final gene models for protein coding genes were identified

using GeneMarkS [16] version 2.6r and Glimmer [29] version 3, with differences between the two methods resolved through hand curation.

The proteins coded by these final models were then annotated using InterProScan [119], CDD batch search [89], and by using BLASTP [3] version 2.2.25+ to search the following protein databases: COG [145], arCOG [88] (2009 update), KEGG [66] (Jan 2011), NCBI ProtClustDB [70] (November 2010), MEROPS [122] release 9.4 and NCBI non-redundant proteins (downloaded April 27, 2011), all with an e-value cutoff of 10^{-5}. Final annotations for each protein were determined by hand, integrating the results of all of the above searches.

4.5 Analysis of the MG-II genome

4.5.1 Euryarchaeal and marine group II Euryarchaeota *phylogenies*

A maximum likelihood phylogeny of a concatenation of 31 conserved archaeal proteins from ten *Euryarchaeota* with available genomes was produced with PHYML [54] version 2.4.4 using the JTT substitution model, based on the concatenated amino acid sequences. The ten archaeal genomes used were: MG-II (`CM001443`), *Aciduliprofundum boonei* T469 (`CP001941`), *Thermoplasma volcanium* GSS1 (`NC_002689`), *Picrophilus torridus* DSM 9790 (`NC_005877`), *Methanosarcina barkeri* str. Fusaro (`NC_007355`), *Archaeoglobus fulgidus* DSM 4304 (`NC_000917`), *Pyrococcus abyssi* GE5 (`NC_000868`), *Methanobacterium lacus* AL-21 (`NC_015216`), *Halobacterium* sp. NRC-1 (`NC_002607`), *Nitrosopumilus maritimus* SCM1 (`CP000866`), *Methanococcus maripaludis* S2 (`NC_005791`). The proteins were selected following Gao and Gupta [47], and were concatenated in the order shown in Tables B.4 and B.5, and aligned with the Geneious Pro [37] version 5.3 global alignment tool using the following settings: (Cost matrix: Blosum62, Gap open penalty: 12, Gap extension penalty: 3, Refinement iterations: 3). The tree is shown rooted with the same concatenation of proteins from *Nitrosopumilus maritimus* (`CP000866`), a member of phylum *Thaumarchaeota*. Branch lengths

are proportional to the number of substitutions per site. Bootstraps are shown for 100 replicate trees. The results are shown in Figure 2.5A.

A maximum likelihood phylogeny of full-length marine group II 16S rRNA sequences was produced with PHYML [54] version 2.4.4 using the JC69 substitution model. 16S rDNA sequences were obtained from the Ribosomal Database Project [26] website except for clones generated from the October clone libraries (this study). The standard RDP alignment was used with October clone library sequences added to that alignment (preserving existing gaps) with the Geneious Pro [37] version 5.3 global alignment tool using the following settings: (Cost matrix: 70% similarity (IUB), Gap open penalty: 12, Gap extension penalty: 3, Refinement iterations: 2, Type: free end gaps). The tree is shown rooted with the 16S rDNA sequence of *Aciduliprofundum boonei*. Branch lengths are proportional to the number of substitutions per site. Bootstraps are shown for 100 replicate trees. The results are shown in Figure A.15.

4.5.2 Alignment of environmental sequences to the MG-II genome

Candidate environmental clones were identified from the top Genbank hits of the MG-II protein BLASTP [3] search of the NCBI nr protein database (see genome annotation methods above). Candidate metagenomic assemblies were identified from the top hits of a MG-II protein BLASTP search of the NCBI env-nr protein database. Corresponding nucleotide sequences for both sets of environmental sequence hits were then retrieved from Genbank.

Each of these sets of environmental sequences were separately aligned with the MG-II genome final assembly using the PROMER tool from the MUMMER [77] software package version 3.22 using alignment parameters `--maxmatch -c 15 -b 100 -g 50`. The show-tiling command was then used with parameters `-a -i 50 -V 0 -v 50 -g 20000` to produce the tiled placements of environmental clones and metagenomic assemblies shown in

Figure 2.3A. Gene model alignments between selected environmental clone sequences and the MG-II genome were made using the Reciprocal Best Hits approach described below, and plotted using a customized version of the 'R' language package genoPlotR [55] version 0.6.1 (Figures 2.3B and A.16).

4.5.3 Proteorhodopsin phylogeny

Selected rhodopsin proteins from sequenced genomes and environmental clones were downloaded from GenBank. Full-length rhodopsin proteins from the NCBI env-nr database were identified using marine group II rhodopsins identified through environmental sequence alignments (Figure 2.3B) as BLASTP [3] version 2.2.25+ query sequences with an e-value cutoff of 10^{-30}, with sequence hits covering less than 75% of query genes rejected. The 80 rhodopsin amino acid sequences selected were aligned with the Geneious Pro [37] version 5.3 global alignment tool using the following settings: (Cost matrix: Blosum62, Gap open penalty: 12, Gap extension penalty: 3, Refinement iterations: 3).

A Bayesian phylogeny of rhodopsin proteins was produced from this alignment with MrBayes [128] version 3.1.2 using a poisson rate matrix, gamma rate variation with 4 categories and the sequence `YP_325981`, the halorhodopsin from *Natronomonas pharaonis*, as the outgroup. Additional MrBayes parameters used were: (Chain length: 1000000, Subsampling Freq: 200, Heated chains: 4, Burn-in length: 100000, Heated chain temp: 0.2, and Unconstrained branch lengths: 10). An unrooted cladogram of the resulting phylogeny is shown in Figure 2.4 and the full tree in Figure A.17. Neighbor-joining and Maximum-likelihood trees made from this alignment (not shown) produced substantially the same topology at all major branches as the above Bayesian analysis.

Syntenous gene models flanking the rhodopsin protein consistent with the MG-II genome assembly were identified using Reciprocal Best Hit (RBH) analysis [125] (described be-

low) of all protein models corresponding to the environmental assembles selected for their rhodopsin proteins above. The positions of environmental models homologous to syntenous MG-II proteins were then hand curated. Results of this analysis are summarized in Figure 2.4 and detailed in Figure A.17.

4.5.4 Reciprocal blast homologies

All protein homology inferences used for genome comparisons and alignment of gene models between environmental sequences and the MG-II genome were produced using the Reciprocal Best Hits (RBH) approach [125]. Analysis was performed with custom software utilizing BLASTP [3] version 2.2.25+ with an e-value cutoff of 10^{-5} for all amino acid searches. Pairwise RBH analyses were performed for seven archaeal genomes: MG-II (this study), *Aciduliprofundum boonei* (NC_013926), *Picrophilus torridus* (NC_005877), *Thermoplasma volcanium* (NC_002689), *Archaeoglobus fulgidus* (NC_000917), *Pyrococcus abyssi* (NC_000868), and *Nitrosopumilus maritimus* (CP000866), see Table B.10 for results.

A 3-way comparison of homologous proteins coded by the MG-II, *A. boonei*, and *N. maritimus* genomes was also performed, with transitivity of RBHs as a requirement for classification in the "core" set of 291 archaeal homologs detected (Figures A.14 and 2.5B). The set of ribosomal proteins in the MG-II and *A. boonei* genomes was cataloged and found to be identical with all syntenous arrangements preserved between the two genomes (Table B.6). "Non-homologous proteins", those without RBHs to either other comparison genome, were identified for the MG-II, *A. boonei*, and *N. maritimus* genomes.

4.5.5 Taxonomic classification of proteins

"Non-homologous" proteins coded by genes found among the MG-II, *A. boonei*, and *N. maritimus* geneomes were used to query the RefSeq subset of the NCBI nr protein database

(downloaded April 27, 2011) using BLASTP [3] version 2.2.25+ with an e-value cutoff of 10^{-5} and a maximum of 500 hits. The software package MEGAN [63] version 4.40.5 was used to assign a taxonomic classification to each "non-homologous" protein based on its lowest common ancestor (LCA) algorithm.

An evaluation of the impact of taxonomic filtering of blast hits on the MEGAN classification results was made for all three genomes (Figure A.18E) and filtering was subsequently used only for the *A. boonei* and *N. maritimus* analyses. For MG-II and *A. boonei*, taxonomic filtering excluded hits to RefSeq proteins from five genomes: *A. boonei*, taxid: `439481`; *Thermoplasma acidophilum*, taxid: `273075`; *Thermoplasma volcanium*, taxid: `273116`; *Picrophilus torridus*, taxid: `263820`; and *Ferroplasma acidarmanus*, taxid: `333146`. For *N. maritimus*, taxonomic filtering excluded hits to proteins from three RefSeq genomes: *N. maritimus*, taxid: `436308`; *Candidatus Nitrosoarchaeum limnia*, taxid: `886738`; and *Cenarchaeum symbosium*, taxid: `414004`.

An evaluation of the effect of three settings (5, 10 and 20%) of the "Top Percent" parameter of MEGAN's LCA algorithm was conducted (Figure A.18F), and the value of 10% selected for all subsequent use. Results of the MEGAN protein classifications are summarized in Figures 2.5C, A.18A-A.18D, and Table B.11. Gene models from the "non-homologous" proteins in the MG-II genome that MEGAN classified within the domain Bacteria are plotted on Figure A.14.

4.6 Assembly of virus genomes

Raw color-space SOLiD reads from the same October and May samples described previously in [64] were converted to the FASTQ format for compatibility with standard tools, de-duplicated to remove redundant PCR duplicated mate-pairs, and trimmed/filtered for quality, all using custom software included with the SEAStAR tools version 0.5 (see Ap-

pendix 3). Conversion to FASTQ was accomplished with the `solid2fastq` tool run with the `-x -z` options. PCR duplicate mate-paired reads were removed using the `fastq_nodup` tool run with options `-z -l 15 -d 2 -e 3`. For trimming we used the `trimfastq` tool, and quality scores were used to determine the cumulative probability of zero uncorrected nucleotide-space errors occurring up to each position in a read. Reads used for alignment from each sequencing run were independently trimmed using an error probability threshold tuned to maximize the sequence aligned with a known reference genome using BWA version 0.5.10 [83]. For the October sample, the genome sequence of *Pelagibacter ubique* str. HTCC1062 (`NC_007205`) was used, yielding a trimming threshold of 0.4. For the May sample, the genome sequence of *Enterobacteria* phage lambda (`NC_001416`, an added standard) was used, yielding a threshold of 0.5. These values correspond with the predicted probability that a trimmed read is free of uncorrectable nucleotide errors, based on instrument generated quality scores. Reads trimmed below 34nt in length were discarded, as were low complexity reads with measured mean entropy of less than 3 bits per di-nucleotide after trimming. Reads containing low complexity sequence were discarded because this is a common error mode for SOLiD generated sequence, and such reads consume memory and complicate *de novo* assembly. The `trimfastq` options implementing the above described settings are:

```
# For the October sample:
trimfastq -z -p 0.4 -l 34 -m 34 --add_len -e 3.0
# For the May sample:
trimfastq -z -p 0.5 -l 34 -m 34 --add_len -e 3.0
```

Quality trimmed reads used for assembly were then error-corrected using the Applied Biosystems SOLiD™ Accuracy Enhancer Tool (SAET) tool version 2.2 (Foster City, CA) with an expected reference length of 200 megabases and parameters `-trustprefix 25 -localrounds 3 -globalrounds 2`. These settings were selected as a reasonable tradeoff

between error correction potential, and memory use and computation time required by the tool. Because this tool requires inputs in the CSFASTA format, two scripts included with the SEAStAR tools `fastq2csfasta.awk` and `fastq_merge_csfasta.awk` were used to feed the SAET tool from FASTQ format files and to merge any error corrected reads back into the FASTQ file format. Prior to assembly, FASTA versions of the FASTQ files were produced using `trimfastq -z -p 0.0 --fasta` which forgoes trimming and produces compressed FASTA files ready for input to the assembler.

4.6.1 De novo contig assembly of metagenomic reads

Color-space contigs were assembled from the SAET error-corrected FASTA read sets from each of the two sample metagenomes using the program Velvet [168] version 1.2.08 with a k-mer sizes of 23, 25, 27, 29 and 31 and parameters:

```
velveth_de 23,32,2 -create_binary -fasta -shortPaired <matesfile> \
    -short <singletsfile>
# And for each k-mer length:
velvetg_de <infile> -scaffolding no -ins_length X -ins_length_sd Y \
    -exp_cov 1000 -cov_cutoff 8 -min_contig_lgth 75
```

where the insert length `X` and standard deviation `Y` were estimated from reference alignments to be `X=1000`, `Y=250` and `X=2100`, `Y=300` for the October and May samples, respectively. All other parameters were selected based on recommendations in the Velvet documentation. Results of these assemblies are summarized in Table B.12.

4.6.2 Re-alignment of reads to contigs, mate-pair processing, visualization and scaffolding

Quality trimmed reads from the each sample metagenome were aligned back to the corresponding color-space assembly contigs (for each k-mer value) using BWA [83] version 0.5.10 as described for the reference databases. Color-space contigs were naively

converted to nucleotide-space using a constant nucleotide prefix (using the SEAStAR `csfasta2ntfasta.awk` script) for the purposes of input to BWA, which requires nucleotide-space reference sequences but immediately converts them back to color-space internally when the `-c` parameter is used. The resulting SAM alignment files were post-processed with the SEAStAR `ref_select` tool to produce nucleotide-space consensus contigs with coverage estimates and bit-scored mate-pair connections between contigs, using parameters `-q -m --mp_share_lim=30.0 --mp_mate_lim=30.0`. These settings filter out all mate-pair connections scoring less than 30 bits.

Briefly, `ref_select` uses a dynamic programming algorithm to generate consensus nucleotide-space contigs from color-space read alignments, and assigns bit-scores to mate-pair connections within and between contig sequences based on the sum of the minimum alignment bit-score (calculated as described above) assigned to each pair of mated reads. `ref_select` emits a mate-pair connection graph which represents contigs as nodes, mate-pair connections as edges, and encodes for each element of the graph, statistics such as total bit-score, coverage, %GC content, sequence length, and mean aligned positions of mate-paired reads.

A full set of linear scaffolds was produced for each sample using the SEAStAR `graph_ops` tool from the 25nt k-mer assemblies using the following subcommands:

```
MST
PLUCK
PRUNE
PUSH
SLICE
INSERT
SCAFF
```

For the October sample, this resulted in 13,948 scaffolds with N50 of 933 and almost 73 Mbp of total sequence with 61-fold mean read coverage. For the May sample, this resulted

in 10,164 scaffolds with N50 of 1261 and almost 72 Mbp of total sequence with 59-fold mean read coverage.

4.6.3 Circular scaffold detection and reassembly

Circular scaffold sequences were detected and output using the SEAStAR `graph_ops` tool for each sample and k-mer length through a series of operations on each of the assembly mate-pair connection graphs. Each graph was transformed into a set of spanning trees with edge weights determined using a heuristic taking into account mate-pair bit-scores and the estimated coverage and %GC content of the connected contigs. In this way, the remaining connections maximized parsimony by eliminating the weakest connections and those between contigs with divergent %GC and coverage estimates. Two different approaches were used for each sample and k-mer combination, the maximal spanning tree `MST` and the scaffold spanning tree `SST` algorithms. The `graph_ops` commands `PLUCK PRUNE PUSH` were run on each assembly to yield parsimonious linear chains of mate-pair connected contigs. Then at three points in the subsequent analysis, a circle detection process implemented by the `graph_ops` `CIRCLE` command was run to produce and output putative circular scaffolds. The three points were:

1. Immediately after the `PUSH` command.
2. After then running the `SLICE` command, to cut scaffolds with anomalous coverage or %GC content.
3. And then after running the `INSERT` and `SCAFF` commands to produce complete scaffolds.

Briefly, the `graph_ops` `CIRCLE` command detects circular scaffolds by looking for mate-pair connections between contigs at or one "hop" into each end of a linear scaffold. If such

a connection exists, then the scaffold is deemed "circular" and is output for subsequent processing. Once this was complete there were a total of 30 sets of output circles for each sample (one for each of five k-mer values, for both the MST and SST algorithms, with circles detected and output at each of three stages described above). For each of the 30 sets of circles, all reads (and their mates) aligning to a contig in any one of the identified circles was output, using the `ref_select` options:

```
ref_select --read_output=<prefix> --read_output_gzip \
     --exclude_file=<contignames> --invert_exclude -q -m \
     --mp_share_lim=0.0 --mp_mate_lim=0.0
```

The result of this operation was a single set of mate-paired sequence reads for each of the October and May samples, where every read was associated with at least one of the circular scaffolds from the 30 subassemblies described above. Each of these limited sets of "circle associated" reads was reassembled with Velvet version 1.2.08 [168] with a k-mer length of 25 and all other parameters as described above for the full metagenome assemblies. From these assemblies, a new and final set of circular scaffolds was produced for the October and May samples using the above described methods, except that the `graph_ops` tool was run with the following subcommands:

```
MST
PLUCK
PRUNE
PUSH
INSERT
SCAFF
SELCC {"min_seqlen":25000,"min_nodes":3}
CIRCLE {"thresh":0.0}
```

The resulting circular scaffolds for October ($n = 59$) and May ($n = 91$) are summarized in Figures A.19 and A.20, and Tables B.13 and B.14.

4.7 Identification and annotation of archaeal virus genomes

Gene models for the circular scaffolds were computed using Glimmer version 3 [29], with a new gene model computed for each scaffold from long ORFs. The Bacteria/Archaea genetic code was used with start codon probabilities `atg=0.6 gtg=0.35 ttg=0.05` and stop codons `tag tga taa`. Glimmer was instructed to use a position weighted matrix, and automatically determined GC %. A minimum gene length of 150nt, max overlap of 0, and threshold score of 30 were used, with genes permitted to be called past the end of the sequence. The October and May circular scaffolds produced 3700 and 6263 gene models, respectively.

A preliminary search for putative Archaeal viruses was conducted by building custom BLAST [3] databases from amino acid translations of the above gene models, and then searching for hits using protein sequences from the marine group II (MG-II) *Euryarchaeote* genome (`CM001443`) and the genome of *Nitrosopumilus maritimus* (`CP000866`). BLASTP version 2.2.25+ was used with a e-value cutoff of 10^{-20}. Initial searches of the May sample indicated that the largest of the "circular" scaffolds was part of a bacterial genome, and another was a section of the MG-II genome previously reconstructed from the same sample [64]. After discarding proteins from these two scaffolds from the BLAST database, searches conducted with the MG-II genome resulted in 24 proteins with hits in the October scaffolds and 44 with hits in the May scaffolds. Similar searches using the *N. maritimus* proteins yielded 28 and 38 proteins with hits for the October and May samples, respectively.

Proteins with hits were examined for those that appeared to be characteristically Archaeal, one protein in particular, the Archaeal thermosome subunits (HSP60 family protein chaperones [81]) stood out. The three thermosome subunits present in the MG-II genome (`MG2_0229`, `MG2_0752` and `MG2_1632`) each hit a protein on three scaffolds in the October sample (scaffolds `004`, `006` and `041`, designated `MEV1`, `MEV4` and `MEV2` respectively) and one scaffold in the May sample (scaffold `010`, designated `MEV5`), see Tables B.13 and B.14. The

two thermosome subunits in the *N. maritimus* genome (Nmar_1577 and Nmar_1792) also hit the same four scaffolds, although with lower scoring hits than the MG-II proteins. Follow-up BLASTP searches using translated genes models from the four circular scaffolds with thermosome hits revealed one more circular scaffold with shared homologous proteins and an HSP60 family protein chaperone (scaffold 023 from the October sample, designated MEV3, see Table B.13). Similarly, BLASTP searches of the full compliment of assembled linear scaffolds from the October and May using translated protein models from the five identified circular scaffolds yielded three additional scaffolds with many high scoring homologous proteins (designated GG2SC877, GG3SC144, and GG2SC16, data not shown).

4.7.1 Gene and function prediction

For each of the eight scaffolds (five circular MEV1-MEV5 and three linear GG2SC877, GG3SC144, and GG2SC16), final gene models for protein coding genes were identified using Glimmer [29] version 3, with the same settings described in the previous section. RNAmmer [80] version 1.2 was also run, but no non-coding RNA genes were identified in any of the sequences.

The protein functions for these final gene models were annotated using results from InterProScan [119], CDD batch search [89], and by using BLASTP [3] version 2.2.25+ to search the following databases: NCBI non-redundant proteins (nr), NCBI environmental nucleotides (env-nt) and NCBI Refseq virus genomes (all downloaded September 28, 2015), with an e-value cutoff of 10^{-5}. Annotations for each protein model were determined by hand, integrating the results of all of the above searches (see Figures A.21, A.22, A.23 and Tables B.16, B.17 and B.18).

4.7.2 Virus reciprocal blast homologies

A candidate environmental fosmid clone from the Mediterranean Sea (AP014343) [96] was identified from the top BLASTP hits [3] to the NCBI nr protein database (see genome annotation methods above) by proteins in the MEV4 and MEV5 scaffolds.

All protein homology inferences used for genome comparisons and alignment of gene models between the circular scaffolds (MEV1-MEV5) and the linear scaffolds (GG2SC877, GG3SC144, and GG2SC16) and fosmid clone (AP014343) produced using the Reciprocal Best Hits (RBH) approach [125]. Analysis was performed with custom software utilizing BLASTP [3] version 2.2.25+ with an e-value cutoff of 10^{-5} for all amino acid searches. Pairwise RBH analyses were performed between each pair of the scaffolds, and it was revealed that scaffolds clustered into three distinct groups correlated with the approximate genome size (Tables 3.1, 3.2, B.15, B.16, B.17 and B.18).

The resulting gene model alignments were plotted by group using a customized version of the 'R' language package genoPlotR [55] version 0.6.1 (Figure 3.2). The environmental fosmid (AP014343) is clearly related to the "Group 3" genomes, although it does not contain a thermosome gene. Based on its length and gene content, it is likely a genome fragment representing approximately 25-30% of the full genome (Figure 3.2C). RBH analysis results among the three groups were also plotted for the three virus groups using MEV1, MEV3 and MEV4 as representative sequences using genoPlotR as described above (Figure 3.3).

4.8 Phylogenies of archaeal virus proteins

4.8.1 Identification of additional assembled thermosome sequences

The MG-II reference genome was originally assembled entirely from the May sample [64]. To build a more complete phylogeny of euryarchaeal thermosome sequences, we used

TBLASTN [3] version 2.2.25+ to search for additional scaffolds assembled from the October and May samples containing thermosome genes, using the three thermosome genes from the MG-II genome (`MG2_0229`, `MG2_0752` and `MG2_1632`) to query the scaffold nucleotide databases using an e-value cutoff of 10^{-20}. These searches resulted in high scoring hits to six scaffolds from the October sample and one additional scaffold (beyond those identical to the MG-II reference genome) from the May sample (see Table B.19).

RBH analyses, as described above, were performed between each of these scaffolds and the MG-II genome to look for syntenic gene ordering around the matching thermosome gene in the reference sequence, and plotted using using genoPlotR as described above (Figures A.27 and A.28). One of the assembled scaffolds from October (`GG2SC40`) contained two predicted thermosome genes, and upon inspection the scaffold was discovered to contain two distinct halves with differing read coverage and GC % statistics. An RBH analysis using the MG-II reference genome and the `MEV4` circular scaffold revealed this assembled scaffold to contain part of a virus sequence (related to the Group 3 virus genomes), attached to a marine group II *Euryarchaeote* genome fragment (Figure A.28). This scaffold sequence may represent one end of a virus genome integrated within the host genome, or it could be an assembly chimera. Manual inspection of the raw mate-pair assembly graph from which this scaffold was created was inconclusive in resolving this question.

4.8.2 Thermosome subunit phylogeny

Selected archaeal thermosome and bacterial and archaeal groEL (HSP60 family protein chaperone) proteins from sequenced genomes and environmental clones were downloaded from GenBank. Full-length environmental proteins from the NCBI nr database were identified using the thermosome proteins from the MG-II genome as BLASTP [3] version 2.2.25+ query sequences with an e-value cutoff of 10^{-20}. The 89 thermosome and groEL amino

acid sequences selected, added to thermosome protein sequences from the eight virus scaffolds and seven additional marine group II scaffolds discussed above, were aligned with the Geneious Pro [37] version 9.0 global alignment tool using the following settings: (Cost matrix: `Blosum62`, Gap open penalty: `12`, Gap extension penalty: `3`, Refinement iterations: `5`).

A Bayesian phylogeny of thermosome and groEL proteins was produced from this alignment with MrBayes [128] version 3.2.2 using a poisson rate matrix, gamma rate variation with 4 categories and the sequence `WP_000729117`, the groEL from *Escherichia coli* K-12, as the outgroup. Additional MrBayes parameters used were: (Chain length: `1100000`, Subsampling Freq: `200`, Heated chains: `4`, Burn-in length: `100000`, Heated chain temp: `0.2`, and Unconstrained branch lengths: `10`). An unrooted cladogram of the resulting phylogeny is shown in Figure 3.5 and the full tree in Figure A.29. Maximum-likelihood trees made from this alignment (not shown) produced substantially the same topology at all major branches as the above Bayesian analysis.

Protein chaperone amino acid sequences identified as coming from marine group III environmental fosmids in Figures 3.5 and A.29 were determined to be such through the process of eliminating members of the marine group II *Euryarchaeota* the source of such proteins, leaving members of the marine group III *Euryarchaeota* as the only likely source given the sampling location in the deep waters of the Mediterranean Sea [36]. Examination of neighboring genes on in the fosmid sequences revealed no synteny with the MG-II genome in the three locations where thermosome genes are found. Further evidence was gathered by examining the phylogenetic relationship of ribosomal proteins found on the same (or closely related) fosmids. For groEL bearing fosmids `AD1000-53-H05`, `SAT1000-24-G08` and `KM3-13-C08`, these sequences are syntenous with each other, and the phylogenetic tree depicted in Figure S2 of [36] clearly shows the ribosomal S2 protein on these fosmids clustering within the *Euryarchaeota*, but separately from the homologous protein in known ma-

rine group II sequences, demonstrating that these groEL genes to be from members of the marine group III *Euryarchaeota*.

Genes on the thermosome bearing fosmids KM3_89_F04, KM3_141_A08, KM3_79_B02 and AD1000_99_D12 are also largely syntenous the each other and all have a copy of the ribosomal S6e protein. Ribosomal protein S6e sequences from the genomes of eleven selected Archaea (including the MG-II genome) were aligned with the S6e sequences from these four fosmids using the Geneious Pro [37] version 9.0 global alignment tool using the following settings: (Cost matrix: Blosum62, Gap open penalty: 12, Gap extension penalty: 3, Refinement iterations: 5). A phylogenetic tree was created from the resulting alignment using FastTree version 2.1.5 [116] using parameters (substitution model: JTT, Rate categories: 20, Optimize Gamma20 likelihood: true). As with the ribosomal S2 protein above, the resulting ribosomal S6e protein phylogeny clearly shows that these sequences are *Euryarchaeal* but do not cluster with the marine group II sequence (Figure A.26). One additional thermosome bearing fosmid sequence AD1000_23_H03 shares gene order synteny with the four listed above, but ends before the region containing the ribosomal S6e gene. However, this fosmid possesses a dehydrogenase gene next to its thermosome (AIE92485) that codes a protein 97% identical to that of an identically positioned gene (AIE97416) on fosmid AD1000_99_D12. Therefore, the five thermosome genes on these fosmids are almost certainly from members of the marine group III *Euryarchaeota*.

4.8.3 Phylogenies of viral DNA replication machinery

Each of the three groups of virus genomes shares homologous proteins predicted to code for DNA replication machinery with one other group. Of particular interest are the DNA polymerase elongation subunit shared by groups 1 and 2 and the DNA sliding clamp subunit shared by groups 2 and 3 (Figure 3.3, Tables 3.2 and B.15). To investigate the relationships

among these proteins and other Archaea and archaeal viruses, phylogenetic trees were constructed each protein.

DNA polymerase elongation subunit proteins were downloaded from GenBank from selected archaeal genomes, including the MG-II genome. Proteins from archaeal viruses were identified from the NCBI nr database using the proteins from the eight viral scaffolds as BLASTP [3] version 2.2.25+ query sequences with an e-value cutoff of 10^{-5}. The 22 amino acid sequences selected, along with five sequences from the virus scaffolds, were aligned with the Geneious Pro [37] version 9.0 global alignment tool using the following settings: (Cost matrix: `Blosum62`, Gap open penalty: `12`, Gap extension penalty: `3`, Refinement iterations: `5`).

DNA polymerase sliding clamp subunit proteins were downloaded from GenBank from selected archaeal genomes, as described above, Proteins from archaeal viruses were also identified as described above. The 14 amino acid sequences selected, along with four sequences from the virus scaffolds, were aligned as described above.

Phylogenetic tree for each of the two DNA polymerase subunit alignments were computed using FastTree version 2.1.5 [116] using parameters (substitution model: `JTT`, Rate categories: `20`, Optimize Gamma20 likelihood: `true`). The resulting phylogenies are depicted in Figures A.24 and A.25.

BIBLIOGRAPHY

[1] Silvia G Acinas, Vanja Klepac-Ceraj, Dana E Hunt, Chanathip Pharino, Ivica Ceraj, Daniel L Distel, and Martin F Polz. Fine-scale phylogenetic architecture of a complex bacterial community. *Nature*, 430(6999):551–554, July 2004.

[2] Eric E Allen and Jillian F Banfield. Community genomics in microbial ecology and evolution. *Nature Reviews Microbiology*, 3(6):489–498, June 2005.

[3] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, September 1997.

[4] Florent E Angly, Ben Felts, Mya Breitbart, Peter Salamon, Robert A Edwards, Craig A Carlson, Amy M Chan, Matthew Haynes, Scott Kelley, Hong Liu, Joseph M Mahaffy, Jennifer E Mueller, Jim Nulton, Robert J Olson, Rachel Parsons, Steve Rayhawk, Curtis A Suttle, and Forest Rohwer. The Marine Viromes of Four Oceanic Regions. *PLoS biology*, 4(11):e368, November 2006.

[5] John M Archibald, Christian Blouin, and W Ford Doolittle. Gene duplication and the evolution of group II chaperonins: Implications for structure and function. *Journal of Structural Biology*, 135(2):157–169, August 2001.

[6] John M Archibald, John M Logsdon, and W Ford Doolittle. Recurrent paralogy in the evolution of archaeal chaperonins. *Current Biology*, 9(18):1053–1056, 1999.

[7] F Azam, T Fenchel, J G Field, J S Gray, L A Meyerreil, and F Thingstad. The Ecological Role Of Water-Column Microbes In The Sea. *Marine Ecology-Progress Series*, 10(3):257–263, 1983.

[8] Farooq Azam and Francesca Malfatti. Microbial structuring of marine ecosystems. *Nature Reviews Microbiology*, 5(10):782–791, October 2007.

[9] Ramy K Aziz, Daniela Bartels, Aaron A Best, Matthew DeJongh, Terrence Disz, Robert A Edwards, Kevin Formsma, Svetlana Gerdes, Elizabeth M Glass, Michael Kubal, Folker Meyer, Gary J Olsen, Robert J Olson, Andrei L Osterman, Ross A Overbeek, Leslie K McNeil, Daniel Paarmann, Tobias Paczian, Bruce Parrello, Gordon D

Pusch, Claudia Reich, Rick Stevens, Olga Vassieva, Veronika Vonstein, Andreas Wilke, and Olga Zagnitko. The RAST Server: rapid annotations using subsystems technology. *BMC genomics*, 9:75, 2008.

[10] Susan M Barns, Charles F Delwiche, Jeffrey D Palmer, and Norman R Pace. Perspectives on archaeal diversity, thermophily and monophyly from environmental rRNA sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 93(17):9188–9193, August 1996.

[11] Elizabeth R Barry and Stephen D Bell. DNA Replication in the Archaea. *Microbiology and Molecular Biology Reviews*, 70(4):876–887, December 2006.

[12] Oded Béjà. To BAC or not to BAC: marine ecogenomics. *Current opinion in biotechnology*, 15(3):187–190, June 2004.

[13] Oded Béjà, L Aravind, Eugene V Koonin, Marcelino T Suzuki, Andrew Hadd, Linh P Nguyen, Stevan B Jovanovich, Cristian M Gates, Robert A Feldman, John L Spudich, Elena N Spudich, and Edward F DeLong. Bacterial rhodopsin: evidence for a new type of phototrophy in the sea. *Science (New York, NY)*, 289(5486):1902–1906, September 2000.

[14] Oded Béjà, Marcelino T Suzuki, Eugene V Koonin, L Aravind, Andrew Hadd, Linh P Nguyen, Rachel Villacorta, Mojgan Amjadi, Corey Garrigues, Stevan B Jovanovich, Robert A Feldman, and Edward F DeLong. Construction and analysis of bacterial artificial chromosome libraries from a marine microbial assemblage. *Environmental microbiology*, 2(5):516–529, October 2000.

[15] Øivind Bergh, Knut Yngve Børsheim, Gunnar Bratbak, and Mikal Heldal. High Abundance of Viruses Found in Aquatic Environments. *Nature*, 340(6233):467–468, 1989.

[16] John Besemer and Mark Borodovsky. GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses. *Nucleic acids research*, 33(Web Server issue):W451–4, July 2005.

[17] Céline Brochier-Armanet, Bastien Boussau, Simonetta Gribaldo, and Patrick Forterre. Mesophilic Crenarchaeota: proposal for a third archaeal phylum, the Thaumarchaeota. *Nature Reviews Microbiology*, 6(3):245–252, March 2008.

[18] Angus Buckling and Paul B Rainey. Antagonistic coevolution between a bacterium and a bacteriophage. *Philosophical transactions of the Royal Society of London Series B, Biological sciences*, 269(1494):931–936, 2002.

[19] Angus Buckling and Paul B Rainey. The role of parasites in sympatric and allopatric host diversification. *Nature*, 420(6915):496–499, 2002.

[20] Isaac K O Cann and Yoshizumi Ishino. Archaeal DNA replication: Identifying the pieces to solve a puzzle. *Genetics*, 152(4):1249–1267, August 1999.

[21] Tonje Castberg, Runar Thyrhaug, Aud Larsen, Ruth-Anne Sandaa, Mikal Heldal, James L Van Etten, and Gunnar Bratbak. Isolation and characterization of a virus that infects Emiliania huxleyi (Haptophyta). *Journal of Phycology*, 38(4):767–774, August 2002.

[22] Bonnie Chaban, Sandy Y M Ng, and Ken F Jarrell. Archaeal habitats - from the extreme to the ordinary. *Canadian Journal of Microbiology*, 52(2):73–116, February 2006.

[23] Sallie W Chisholm, Sheila L Frankel, Ralf Goericke, Robert J Olson, Brian Palenik, John B Waterbury, Lisa West-Johnsrud, and Erik R Zettler. Prochlorococcus marinus nov. gen. nov. sp.: an oxyphototrophic marine prokaryote containing divinyl chlorophyll a and b. *Archives Of Microbiology*, 157(3):297–300, 1992.

[24] Sallie W Chisholm, Robert J Olson, Erik R Zettler, Ralf Goericke, John B Waterbury, and Nicholas A Welschmeyer. A novel free-living prochlorophyte abundant in the oceanic euphotic zone. *Nature*, 334(6180):340–343, 1988.

[25] D K Clare, P J Bakkes, H van Heerikhuizen, S M van der Vies, and H R Saibil. Chaperonin complex with a newly folded protein encapsulated in the folding chamber. *Nature*, 457(7225):107–110, January 2009.

[26] J R Cole, Q Wang, E Cardenas, J Fish, B Chai, R J Farris, A S Kulam-Syed-Mohideen, D M McGarrell, T L Marsh, G M Garrity, and J M Tiedje. The Ribosomal Database Project: improved alignments and new tools for rRNA analysis. *Nucleic acids research*, 37(Database issue):D141–5, 2009.

[27] Stephanie A Connon and Stephen J Giovannoni. High-throughput methods for culturing microorganisms in very-low-nutrient media yield diverse new marine isolates. *Applied And Environmental Microbiology*, 68(8):3878–3885, August 2002.

[28] R Dawkins and J R Krebs. Arms Races between and within Species. *Proceedings of the Royal Society B: Biological Sciences*, 205(1161):489–511, September 1979.

[29] Arthur L Delcher, Kirsten A Bratke, Edwin C Powers, and Steven L Salzberg. Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics (Oxford, England)*, 23(6):673–679, March 2007.

[30] Edward F DeLong. Archaea in coastal marine environments. *Proceedings of the National Academy of Sciences of the United States of America*, 89(12):5685–5689, June 1992.

[31] Edward F DeLong. Everything in moderation: archaea as 'non-extremophiles'. *Current opinion in genetics & development*, 8(6):649–654, December 1998.

[32] Edward F DeLong. Microbial community genomics in the ocean. *Nature Reviews Microbiology*, 3(6):459–469, June 2005.

[33] Edward F DeLong. The microbial ocean from genomes to biomes. *Nature*, 459(7244):200–206, May 2009.

[34] Edward F DeLong, Christina M Preston, Tracy J Mincer, Virginia Rich, Steven J Hallam, Niels-Ulrik Frigaard, Asuncion Martinez, Matthew B Sullivan, Robert A Edwards, Beltran Rodriguez Brito, Sallie W Chisholm, and David M Karl. Community genomics among stratified microbial assemblages in the ocean's interior. *Science (New York, NY)*, 311(5760):496–503, January 2006.

[35] Ann Demogines, Jonathan Abraham, Hyeryun Choe, Michael Farzan, and Sara L Sawyer. Dual Host-Virus Arms Races Shape an Essential Housekeeping Protein. *PLoS biology*, 11(5), May 2013.

[36] Philippe Deschamps, Yvan Zivanovic, David Moreira, Francisco Rodríguez-Valera, and Purificación López-García. Pangenome Evidence for Extensive Interdomain Horizontal Transfer Affecting Lineage Core and Shell Genes in Uncultured Planktonic Thaumarchaeota and Euryarchaeota. *Genome Biology and Evolution*, 6(7):1549–1563, July 2014.

[37] A J Drummond, B Ashton, S Buxton, M Cheung, A Cooper, C Duran, M Field, J Heled, M Kearse, S Markowitz, R Moir, S Stones-Havas, S Sturrock, T Thierer, and A Wilson. Geneious v5.3.

[38] Paul G Falkowski, Tom Fenchel, and Edward F DeLong. The microbial engines that drive Earth's biogeochemical cycles. *Science (New York, NY)*, 320(5879):1034–1039, May 2008.

[39] Christopher B Field, Michael J Behrenfeld, James T Randerson, and Paul Falkowski. Primary production of the biosphere: Integrating terrestrial and oceanic components. *Science (New York, NY)*, 281(5374):237–240, 1998.

[40] Niels-Ulrik Frigaard, Asuncion Martinez, Tracy J Mincer, and Edward F DeLong. Proteorhodopsin lateral gene transfer between marine planktonic Bacteria and Archaea. *Nature*, 439(7078):847–850, February 2006.

[41] Jed A Fuhrman. Marine viruses and their biogeochemical and ecological effects. *Nature*, 399(6736):541–548, June 1999.

[42] Jed A Fuhrman and Alison A Davis. Widespread archaea and novel bacteria from the deep sea as shown by 16S rRNA gene sequences. *Marine Ecology-Progress Series*, 150(1-3):275–285, 1997.

[43] Jed A Fuhrman, Ian Hewson, Michael S Schwalbach, Joshua A Steele, Mark V Brown, and Shahid Naeem. Annually reoccurring bacterial communities are predictable from ocean conditions. *Proceedings of the National Academy of Sciences of the United States of America*, 103(35):13104–13109, August 2006.

[44] Jed A Fuhrman, Kirk McCallum, and Alison A Davis. Novel major archaebacterial group from marine plankton. *Nature*, 356(6365):148–149, March 1992.

[45] Pierre E Galand, Carmen Gutiérrez-Provecho, Ramon Massana, Josep M Gasol, and Emilio O Casamayor. Inter-annual recurrence of archaeal assemblages in the coastal NW Mediterranean Sea (Blanes Bay Microbial Observatory). *Limnology and Oceanography*, 55(5):2117–2125, 2010.

[46] Emden R Gansner and Stephen C North. An open graph visualization system and its applications to software engineering. *Software-Practice & Experience*, 30(11):1203–1233, 2000.

[47] Beile Gao and Radhey S Gupta. Phylogenomic analysis of proteins that are distinctive of Archaea and its main subgroups and the origin of methanogenesis. *BMC genomics*, 8:86, 2007.

[48] C Geslin, M Gaillard, D Flament, K Rouault, M Le Romancer, D Prieur, and G Erauso. Analysis of the first genome of a hyperthermophilic marine virus-like particle, PAV1, isolated from Pyrococcus abyssi. *Journal Of Bacteriology*, 189(12):4510–4519, June 2007.

[49] Rohit Ghai, Ana-Belen Martin-Cuadrado, Aitor Gonzaga Molto, Inmaculada García Heredia, Raúl Cabrera, Javier Martin, Miguel Verdú, Philippe Deschamps, David Moreira, Purificación López-García, Alex Mira, and Francisco Rodríguez-Valera. Metagenome of the Mediterranean deep chlorophyll maximum studied by direct and fosmid library 454 pyrosequencing. *The ISME journal*, 4(9):1154–1166, September 2010.

[50] Steven R Gill, Mihai Pop, Robert T Deboy, Paul B Eckburg, Peter J Turnbaugh, Buck S Samuel, Jeffrey I Gordon, David A Relman, Claire M Fraser-Liggett, and Karen E Nelson. Metagenomic analysis of the human distal gut microbiome. *Science (New York, NY)*, 312(5778):1355–1359, June 2006.

[51] Stephen J Giovannoni, Theresa B Britschgi, Craig L Moyer, and Katherine G Field. Genetic diversity in Sargasso Sea bacterioplankton. *Nature*, 345(6270):60–63, May 1990.

[52] Stephen J Giovannoni, Harry James Tripp, Scott A Givan, Mircea Podar, Kevin L Vergin, Damon Baptista, Lisa Bibbs, Jonathan Eads, Toby H Richardson, Michiel Noordewier, Michael S Rappé, Jay M Short, James C Carrington, and Eric J Mathur. Genome streamlining in a cosmopolitan oceanic bacterium. *Science (New York, NY)*, 309(5738):1242–1245, August 2005.

[53] Pedro Gomez and Angus Buckling. Bacteria-Phage Antagonistic Coevolution in Soil. *Science (New York, NY)*, 332(6025):106–109, 2011.

[54] Stéphane Guindon and Olivier Gascuel. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 52(5):696–704, October 2003.

[55] Lionel Guy, Jens Roat Kultima, and Siv G E Andersson. genoPlotR: comparative gene and genome visualization in R. *Bioinformatics (Oxford, England)*, 26(18):2334–2335, September 2010.

[56] Jo Handelsman. Metagenomics: Application of genomics to uncultured microorganisms. *Microbiology and Molecular Biology Reviews*, 68(4):669–685, December 2004.

[57] Mark Hedglin, Ravindra Kumar, and Stephen J Benkovic. Replication Clamps and Clamp Loaders. *Cold Spring Harbor Perspectives in Biology*, 5(4), April 2013.

[58] Lydie Herfort, Stefan Schouten, Ben Abbas, Marcel J W Veldhuis, Marco J L Coolen, Cornelia Wuchter, Jan P Boon, Gerhard J Herndl, and Jaap S Sinninghe Damsté. Variations in spatial and temporal distribution of Archaea in the North Sea in relation to environmental variables. *FEMS microbiology ecology*, 62(3):242–257, 2007.

[59] Matthias Hess, Alexander Sczyrba, Rob Egan, Tae-Wan Kim, Harshal Chokhawala, Gary Schroth, Shujun Luo, Douglas S Clark, Feng Chen, Tao Zhang, Roderick I Mackie, Len A Pennacchio, Susannah G Tringe, Axel Visel, Tanja Woyke, Zhong Wang, and Edward M Rubin. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science (New York, NY)*, 331(6016):463–467, January 2011.

[60] Elizabeth A Holland, Frank J Dentener, Bobby H Braswell, and James M Sulzman. Contemporary and pre-industrial global reactive nitrogen budgets. *Biogeochemistry*, 46(1-3):7–43, 1999.

[61] Karin Holmfeldt, Natalie Solonenko, Manesh Shah, Kristen Corrier, Lasse Riemann, Nathan C Verberkmoes, and Matthew B Sullivan. Twelve previously unknown phage genera are ubiquitous in global oceans. *Proceedings of the ...*, 2013.

[62] Arthur L Horwich and Helen R Saibil. The thermosome: chaperonin with a built-in lid. 5(5):333–336, May 1998.

[63] Daniel H Huson, Alexander F Auch, Ji Qi, and Stephan C Schuster. MEGAN analysis of metagenomic data. *Genome Research*, 17(3):377–386, March 2007.

[64] Vaughn Iverson, Robert M Morris, Christian D Frazar, Chris T Berthiaume, Rhonda L Morales, and E Virginia Armbrust. Untangling Genomes from Metagenomes: Revealing an Uncultured Class of Marine Euryarchaeota. *Science (New York, NY)*, 335(6068):587–590, February 2012.

[65] Vaughn S Iverson and Eve A Riskin. A fast method for combining palettes of color quantized images. In *Proceedings of ICASSP '93*, pages 317–320 vol.5. IEEE, 1993.

[66] Minoru Kanehisa, Susumu Goto, Masahiro Hattori, Kiyoko F Aoki-Kinoshita, Masumi Itoh, Shuichi Kawashima, Toshiaki Katayama, Michihiro Araki, and Mika Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic acids research*, 34(Database issue):D354–7, January 2006.

[67] David M Karl. Microbial oceanography: paradigms, processes and promise. *Nature Reviews Microbiology*, 5(10):759–769, October 2007.

[68] Markus B Karner, Edward F DeLong, and David M Karl. Archaeal dominance in the mesopelagic zone of the Pacific Ocean. *Nature*, 409(6819):507–510, January 2001.

[69] David L Kirchman. *Microbial ecology of the oceans*. Hoboken, N.J. : Wiley-Blackwell, 2008.

[70] William Klimke, Richa Agarwala, Azat Badretdin, Slava Chetvernin, Stacy Ciufo, Boris Fedorov, Boris Kiryutin, Kathleen O'Neill, Wolfgang Resch, Sergei Resenchuk, Susan Schafer, Igor Tolstoy, and Tatiana Tatusova. The National Center for Biotechnology Information's Protein Clusters Database. *Nucleic acids research*, 37(Database issue):D216–23, 2009.

[71] Daniel Klunker, Bernd Haas, Angela Hirtreiter, Luis Figueiredo, Dean J Naylor, Günter Pfeifer, Volker Müller, Uwe Deppenmeier, Gerhard Gottschalk, F Ulrich Hartl, and Manajit Hayer-Hartl. Coexistence of group I and group II chaperonins in the archaeon Methanosarcina mazei. *The Journal of biological chemistry*, 278(35):33256–33267, 2003.

[72] Stefan Knapp, Ingeborg Schmidt-Krey, Hans Hebert, Thomas Bergman, Hans Jörnvall, and Rudolf Ladenstein. The Molecular Chaperonin Tf55 From the Thermophilic Archaeon Sulfolobus-Solfataricus - a Biochemical and Structural Characterization. *Journal of molecular biology*, 242(4):397–407, 1994.

[73] Yosuke Koga and Hiroyuki Morii. Biosynthesis of ether-type polar lipids in archaea and evolutionary considerations. *Microbiology and Molecular Biology Reviews*, 71(1):97–120, March 2007.

[74] Martin Könneke, Anne E Bernhard, José R de la Torre, Christopher B Walker, John B Waterbury, and David A Stahl. Isolation of an autotrophic ammonia-oxidizing marine archaeon. *Nature*, 437(7058):543–546, September 2005.

[75] Mart Krupovic, Anja Spang, Simonetta Gribaldo, Patrick Forterre, and Christa Schleper. A thaumarchaeal provirus testifies for an ancient association of tailed viruses with archaea. *Biochemical Society transactions*, 39(1):82–88, February 2011.

[76] Lidia P Kurochkina, Pavel I Semenyuk, Victor N Orlov, Johan Robben, Nina N Sykilinda, and Vadim V Mesyanzhinov. Expression and functional characterization of the first bacteriophage-encoded chaperonin. *Journal of virology*, July 2012.

[77] Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):R12, 2004.

[78] Andrew R Kusmierczyk and Jörg Martin. Chaperonins - keeping a lid on folding proteins. *FEBS letters*, 505(3):343–347, 2001.

[79] Jessica M Labonté, Brandon K Swan, Bonnie Poulos, Haiwei Luo, Sergey Koren, Steven J Hallam, Matthew B Sullivan, Tanja Woyke, K Eric Wommack, and Ramunas Stepanauskas. Single-cell genomics-based analysis of virus–host interactions in marine surface bacterioplankton. *The ISME journal*, April 2015.

[80] Karin Lagesen, Peter Hallin, Einar Andreas Rødland, Hans-Henrik Staerfeldt, Torbjørn Rognes, and David W Ussery. RNAmmer: consistent and rapid annotation of ribosomal RNA genes. *Nucleic acids research*, 35(9):3100–3108, 2007.

[81] Andrew T Large, Martin D Goldberg, and Peter A Lund. Chaperones and protein folding in the archaea. *Biochemical Society transactions*, 37:46–51, February 2009.

[82] Roger S Lasken and Timothy B Stockwell. Mechanism of chimera formation during the Multiple Displacement Amplification reaction. *BMC biotechnology*, 7:19, 2007.

[83] Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, 26(5):589–595, March 2010.

[84] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and Subgroup, 1000 Genome Project Data Processing. The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, 25(16):2078–2079, 2009.

[85] Heng Li, Jue Ruan, and Richard Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, November 2008.

[86] Zong Lin and Hays S Rye. GroEL-Mediated protein folding: Making the impossible, possible. *Critical Reviews In Biochemistry And Molecular Biology*, 41(4):211–239, 2006.

[87] Wolfgang Ludwig, Oliver Strunk, Ralf Westram, Lothar Richter, Harald Meier, Yadhukumar, Arno Buchner, Tina Lai, Susanne Steppi, Gangolf Jobb, Wolfram Förster, Igor Brettske, Stefan Gerber, Anton W Ginhart, Oliver Gross, Silke Grumann, Stefan Hermann, Ralf Jost, Andreas König, Thomas Liss, Ralph Lüssmann, Michael May, Björn Nonhoff, Boris Reichel, Robert Strehlow, Alexandros Stamatakis, Norbert Stuckmann, Alexander Vilbig, Michael Lenke, Thomas Ludwig, Arndt Bode, and Karl-Heinz Schleifer. ARB: a software environment for sequence data. *Nucleic acids research*, 32(4):1363–1371, 2004.

[88] Kira S Makarova, Alexander V Sorokin, Pavel S Novichkov, Yuri I Wolf, and Eugene V Koonin. Clusters of orthologous genes for 41 archaeal genomes and implications for evolutionary genomics of archaea. *Biology direct*, 2:33, 2007.

[89] Aron Marchler-Bauer, Shennan Lu, John B Anderson, Farideh Chitsaz, Myra K Derbyshire, Carol DeWeese-Scott, Jessica H Fong, Lewis Y Geer, Renata C Geer, Noreen R Gonzales, Marc Gwadz, David I Hurwitz, John D Jackson, Zhaoxi Ke, Christopher J Lanczycki, Fu Lu, Gabriele H Marchler, Mikhail Mullokandov, Marina V Omelchenko, Cynthia L Robertson, James S Song, Narmada Thanki, Roxanne A Yamashita, Dachuan Zhang, Naigong Zhang, Chanjuan Zheng, and Stephen H Bryant. CDD: a Conserved Domain Database for the functional annotation of proteins. *Nucleic acids research*, 39(Database issue):D225–9, January 2011.

[90] Sergio Marco, Dionisio Ureña, José L Carrascosa, Thomas Waldmann, Jürgen Peters, Reiner Hegerl, Günter Pfeifer, Hilde Sack-Kongehl, and Wolfgang Baumeister. The Molecular Chaperone Tf55 - Assessment of Symmetry. *FEBS letters*, 341(2-3):152–155, 1994.

[91] Elaine R Mardis. The impact of next-generation sequencing technology on genetics. *Trends in genetics : TIG*, 24(3):133–141, March 2008.

[92] Héctor García Martín, Natalia N Ivanova, Victor Kunin, Falk Warnecke, Kerrie W Barry, Alice Carolyn McHardy, Christine Yeates, Shaomei He, Asaf A Salamov, Ernest Szeto, Eileen Dalin, Nicholas H Putnam, Harris J Shapiro, Jasmyn L Pangilinan, Isidore Rigoutsos, Nikos C Kyrpides, Linda Louise Blackall, Katherine D McMahon, and Philip Hugenholtz. Metagenomic analysis of two enhanced biological phosphorus removal (EBPR) sludge communities. *Nature biotechnology*, 24(10):1263–1269, October 2006.

[93] Ana-Belen Martin-Cuadrado, Inmaculada Garcia-Heredia, Aitor Gonzaga Molto, Rebeca Lopez-Ubeda, Nikole Kimes, Purificación López-García, David Moreira, and Francisco Rodríguez-Valera. A new class of marine Euryarchaeota group II from the mediterranean deep chlorophyll maximum. *International Journal Of Systematic And Evolutionary Microbiology*, 9(7):1619–1634, July 2015.

[94] Anna-Belen Martin-Cuadrado, Francisco Rodríguez-Valera, David Moreira, José C Alba, Elena Ivars-Martínez, Matthew R Henn, Emmanuel Talla, and Purificación López-García. Hindsight in the relative abundance, metabolic potential and genome dynamics of uncultivated marine archaea from comparative metagenomic analyses of bathypelagic plankton of different oceanic regions. *The ISME journal*, 2(8):865–886, August 2008.

[95] Ramon Massana, A E Murray, Christina M Preston, and Edward F DeLong. Vertical distribution and phylogenetic characterization of marine planktonic Archaea in the Santa Barbara Channel. *Applied And Environmental Microbiology*, 63(1):50–56, 1997.

[96] Carolina Megumi Mizuno, Francisco Rodríguez-Valera, Nikole E Kimes, and Rohit Ghai. Expanding the Marine Virosphere Using Metagenomics. *PLoS genetics*, 9(12):e1003987, December 2013.

[97] Gordon E Moore. Cramming More Components Onto Integrated Circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[98] David Moreira, Francisco Rodríguez-Valera, and Purificación López-García. Analysis of a genome fragment of a deep-sea uncultivated Group II euryarchaeote containing 16S rDNA, a spectinomycin-like operon and several energy metabolism genes. *Environmental microbiology*, 6(9):959–969, September 2004.

[99] Robert M Morris, Brook L Nunn, Christian Frazar, David R Goodlett, Ying S Ting, and Gabrielle Rocap. Comparative metaproteomics reveals ocean-scale shifts in microbial nutrient utilization and energy transduction. *The ISME journal*, 4(5):673–685, May 2010.

[100] Robert M Morris, Michael S Rappé, Stephanie A Connon, Kevin L Vergin, William A Siebold, Craig A Carlson, and Stephen J Giovannoni. SAR11 clade dominates ocean surface bacterioplankton communities. *Nature*, 420(6917):806–810, 2002.

[101] Keizo Nagasaki, Yuji Tomaru, Noriaki Katanozaka, Yoko Shirai, Kensho Nishida, Shigeru Itakura, and Mineo Yamaguchi. Isolation and Characterization of a Novel Single-Stranded RNA Virus Infecting the Bloom-Forming Diatom Rhizosolenia setigera. *Applied And Environmental Microbiology*, 70(2):704–711, February 2004.

[102] David M Nelson, Paul Tréguer, Mark A Brzezinski, Aude Leynaert, and Bernard Queguiner. Production and dissolution of biogenic silica in the ocean: Revised global estimates, comparison with regional data and relationship to biogenic sedimentation. *Global Biogeochemical Cycle*, 9(3):359–372, 1995.

[103] Michael Nitsch, Martin Klumpp, Andrei Lupas, and Wolfgang Baumeister. The thermosome: alternating alpha and beta-subunits within the chaperonin of the archaeon Thermoplasma acidophilum. *Journal of molecular biology*, 267(1):142–149, March 1997.

[104] Gary J Olsen, David J Lane, Stephen J Giovannoni, Norman R Pace, and David A Stahl. Microbial ecology and evolution: a ribosomal RNA approach. *Annual review of microbiology*, 40:337–365, 1986.

[105] Norman R Pace. A molecular view of microbial diversity and the biosphere. *Science (New York, NY)*, 276(5313):734–740, May 1997.

[106] Norman R Pace, David A Stahl, David J Lane, and Gary J Olsen. The analysis of natural microbial populations by rRNA sequences. *Adv. Microb. Ecol.*, 9:1–55, 1986.

[107] Annelie Pernthaler, Christina M Preston, Jakob Pernthaler, Edward F DeLong, and Rudolf Amann. Comparison of fluorescently labeled oligonucleotide and polynucleotide probes for the detection of pelagic marine bacteria and archaea. *Applied And Environmental Microbiology*, 68(2):661–667, February 2002.

[108] L E Petrovskaya, E P Lukashev, V V Chupin, S V Sychev, E N Lyukmanova, E A Kryukova, R H Ziganshin, E V Spirina, E M Rivkina, R A Khatypov, L G Erokhina, D A Gilichinsky, V A Shuvalov, and M P Kirpichnikov. Predicted bacteriorhodopsin from

Exiguobacterium sibiricum is a functional proton pump. *FEBS letters*, 584(19):4193–4196, October 2010.

[109] Erik Pettersson, Joakim Lundeberg, and Afshin Ahmadian. Generations of sequencing technologies. *Genomics*, 93(2):105–111, February 2009.

[110] Barry M Phipps, Angelika Hoffmann, Karl O Stetter, and Wolfgang Baumeister. A Novel Atpase Complex Selectively Accumulated Upon Heat-Shock Is a Major Cellular-Component of Thermophilic Archaebacteria. *Embo Journal*, 10(7):1711–1722, July 1991.

[111] Maija K Pietila, Tatiana A Demina, Nina S Atanasova, Hanna M Oksanen, and Dennis H Bamford. Archaeal viruses and bacteriophages: comparisons and contrasts. *Trends in microbiology*, 22(6):334–344, June 2014.

[112] Maija K Pietila, Pasi Laurinmaki, Daniel A Russell, Ching-Chung Ko, Deborah Jacobs-Sera, Sarah J Butcher, Dennis H Bamford, and Roger W Hendrix. Insights into Head-Tailed Viruses Infecting Extremely Halophilic Archaea. *Journal of virology*, 87(6):3248–3260, March 2013.

[113] John M Pisciotta, Yongjin Zou, and Ilia V Baskakov. Light-dependent electrogenic activity of cyanobacteria. *Plos One*, 5(5):e10821, 2010.

[114] Lawrence R Pomeroy. The Ocean's Food Web, A Changing Paradigm. *BioScience*, 24(9):499–504, 1974.

[115] David Prangishvili. The Wonderful World of Archaeal Viruses. *Annual review of microbiology*, 67(1):565–585, September 2013.

[116] Morgan N Price, Paramvir S Dehal, and Adam P Arkin. FastTree 2-Approximately Maximum-Likelihood Trees for Large Alignments. *Plos One*, 5(3), 2010.

[117] Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(Database issue):D61–5, January 2007.

[118] Junjie Qin, Ruiqiang Li, Jeroen Raes, Manimozhiyan Arumugam, Kristoffer Solvsten Burgdorf, Chaysavanh Manichanh, Trine Nielsen, Nicolas Pons, Florence Levenez, Takuji Yamada, Daniel R Mende, Junhua Li, Junming Xu, Shaochuan Li, Dongfang Li, Jianjun Cao, Bo Wang, Huiqing Liang, Huisong Zheng, Yinlong Xie, Julien Tap, Patricia Lepage, Marcelo Bertalan, Jean-Michel Batto, Torben Hansen, Denis Le Paslier,

Allan Linneberg, H Bjørn Nielsen, Eric Pelletier, Pierre Renault, Thomas Sicheritz-Ponten, Keith Turner, Hongmei Zhu, Chang Yu, Shengting Li, Min Jian, Yan Zhou, Yingrui Li, Xiuqing Zhang, Songgang Li, Nan Qin, Huanming Yang, Jian Wang, Søren Brunak, Joel Doré, Francisco Guarner, Karsten Kristiansen, Oluf Pedersen, Julian Parkhill, Jean Weissenbach, MetaHIT Consortium, Peer Bork, and S Dusko Ehrlich. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, March 2010.

[119] E Quevillon, V Silventoinen, S Pillai, N Harte, N Mulder, R Apweiler, and R Lopez. InterProScan: protein domains identifier. *Nucleic acids research*, 33(Web Server issue):W116–20, July 2005.

[120] Michael S Rappé, Stephanie A Connon, Kevin L Vergin, and Stephen J Giovannoni. Cultivation of the ubiquitous SAR11 marine bacterioplankton clade. *Nature*, 418(6898):630–633, August 2002.

[121] Michael S Rappé and Stephen J Giovannoni. The uncultured microbial majority. *Annual review of microbiology*, 57:369–394, 2003.

[122] Neil D Rawlings, Alan J Barrett, and Alex Bateman. MEROPS: the peptidase database. *Nucleic acids research*, 38(Database issue):D227–33, 2010.

[123] A-L Reysenbach and G E Flores. Electron microscopy encounters with unusual thermophiles helps direct genomic analysis of Aciduliprofundum boonei. *Geobiology*, 6(3):331–336, June 2008.

[124] Christian S Riesenfeld, Patrick D Schloss, and Jo Handelsman. Metagenomics: genomic analysis of microbial communities. *Annual review of genetics*, 38:525–552, 2004.

[125] Maria C Rivera, Ravi Jain, Jonathan Moore, and James A Lake. Genomic evidence for two functionally distinct gene classes. *Proceedings of the National Academy of Sciences of the United States of America*, 95(11):6239–6244, May 1998.

[126] Gabrielle Rocap, Frank W Larimer, Jane Lamerdin, Stephanie Malfatti, Patrick Chain, Nathan A Ahlgren, Andrae Arellano, Maureen L Coleman, Loren Hauser, Wolfgang R Hess, Zackary I Johnson, Miriam Land, Debbie Lindell, Anton F Post, Warren Regala, Manesh B Shah, Stephanie L Shaw, Claudia Steglich, Matthew B Sullivan, Claire S Ting, Andrew Tolonen, Eric A Webb, Erik R Zinser, and Sallie W Chisholm. Genome divergence in two Prochlorococcus ecotypes reflects oceanic niche differentiation. *Nature*, 424(6952):1042–1047, August 2003.

[127] Forest Rohwer, Anca Segall, Grieg Steward, Victor Seguritan, Mya Breitbart, Felise Wolven, and Farooq Azam. The complete genomic sequence of the marine phage Roseophage SIO1 shares homology with nonmarine phages. *Limnology and Oceanography*, 45(2):408–418, 2000.

[128] Fredrik Ronquist and John P Huelsenbeck. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics (Oxford, England)*, 19(12):1572–1574, August 2003.

[129] Andreas Ruepp, Werner Graml, Martha-Leticia Santos-Martinez, Kristin K Koretke, Craig Volker, H Werner Mewes, Dmitrij Frishman, Susanne Stocker, Andrei N Lupas, and Wolfgang Baumeister. The genome sequence of the thermoacidophilic scavenger Thermoplasma acidophilum. *Nature*, 407(6803):508–513, September 2000.

[130] Douglas B Rusch, Aaron L Halpern, Granger Sutton, Karla B Heidelberg, Shannon J Williamson, Shibu Yooseph, Dongying Wu, Jonathan A Eisen, Jeff M Hoffman, Karin Remington, Karen Beeson, Bao Tran, Hamilton O Smith, Holly Baden-Tillson, Clare Stewart, Joyce Thorpe, Jason Freeman, Cynthia Andrews-Pfannkoch, Joseph E Venter, Kelvin Li, Saul A Kravitz, John F Heidelberg, Terry Utterback, Yu-Hui Rogers, Luisa I Falcón, Valeria Souza, Germán Bonilla-Rosso, Luis E Eguiarte, David M Karl, Shubha Sathyendranath, Trevor Platt, Eldredge Bermingham, Victor Gallardo, Giselle Tamayo-Castillo, Michael R Ferrari, Robert L Strausberg, Kenneth Nealson, Robert Friedman, Marvin Frazier, and J Craig Venter. The Sorcerer II Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific. *PLoS biology*, 5(3):e77, March 2007.

[131] Douglas B Rusch, Adam C Martiny, Christopher L Dupont, Aaron L Halpern, and J Craig Venter. Characterization of Prochlorococcus clades from iron-depleted oceanic regions. *Proceedings of the National Academy of Sciences of the United States of America*, 107(37):16184–16189, September 2010.

[132] F Sanger, S Nicklen, and A R Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 74(12):5463–5467, December 1977.

[133] Rekha Seshadri, Saul A Kravitz, Larry Smarr, Paul Gilna, and Marvin Frazier. CAMERA: a community resource for metagenomics. *PLoS biology*, 5(3):e75, March 2007.

[134] C E Shannon. A mathematical theory of communication, Part I. *Bell Syst. Tech. J.*, 27:379–423, 1948.

[135] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature biotechnology*, 26(10):1135–1145, October 2008.

[136] Jay Shendure, Robi D Mitra, Chris Varma, and George M Church. Advanced sequencing technologies: methods and goals. *Nature reviews Genetics*, 5(5):335–344, May 2004.

[137] Saskia L Smits, Rogier Bodewes, Aritz Ruiz-Gonzalez, Wolfgang Baumgärtner, Marion P Koopmans, Albert D M E Osterhaus, and Anita C Schürch. Assembly of viral genomes from metagenomes. *Frontiers in Microbiology*, 5, 2014.

[138] Jamie C Snyder, Benjamin Bolduc, and Mark J Young. 40 Years of archaeal virology: Expanding viral diversity. *Virology*, 479:369–378, May 2015.

[139] R Staden. A strategy of DNA sequencing employing computer programs. *Nucleic acids research*, 6(7):2601–2610, June 1979.

[140] James T Staley and Allan Konopka. Measurement of in situ activities of nonphotosynthetic microorganisms in aquatic and terrestrial habitats. *Annual review of microbiology*, 39:321–346, 1985.

[141] Jefferey L Stein, Terence L Marsh, Ke Ying Wu, Hiroaki Shizuya, and Edward F DeLong. Characterization of uncultivated prokaryotes: isolation and analysis of a 40-kilobase-pair genome fragment from a planktonic marine archaeon. *Journal Of Bacteriology*, 178(3):591–599, February 1996.

[142] Ramunas Stepanauskas and Michael E Sieracki. Matching phylogeny and metabolism in the uncultured marine bacteria, one cell at a time. *Proceedings of the National Academy of Sciences of the United States of America*, 104(21):9052–9057, May 2007.

[143] Matthew B Sullivan, John B Waterbury, and Sallie W Chisholm. Cyanophages infecting the oceanic cyanobacterium Prochlorococcus. *Nature*, 424(6952):1047–1051, August 2003.

[144] Curtis A Suttle. Marine viruses — major players in the global ecosystem. *Nature Reviews Microbiology*, 5(10):801–812, October 2007.

[145] Roman L Tatusov, Michael Y Galperin, Darren A Natale, and Eugene V Koonin. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic acids research*, 28(1):33–36, 2000.

[146] Hanno Teeling, Anke Meyerdierks, Margarete Bauer, Rudolf Amann, and Frank Oliver Glöckner. Application of tetranucleotide frequencies for the assignment of genomic fragments. *Environmental microbiology*, 6(9):938–947, September 2004.

[147] Thomas D Mullins, Theresa B Britschgi, Robin L Krest, and Stephen J Giovannoni. Genetic Comparisons Reveal The Same Unknown Bacterial Lineages In Atlantic And Pacific Bacterioplankton Communities. *Limnology and Oceanography*, 40(1):148–158, 1995.

[148] Yuji Tomaru, Kensuke Toyoda, Kei Kimura, Naotsugu Hata, Mikihide Yoshida, and Keizo Nagasaki. First evidence for the existence of pennate diatom viruses. 6(7):1445–1448, January 2012.

[149] Alexander H Treusch, Sven Leininger, Arnulf Kletzin, Stephan C Schuster, Hans-Peter Klenk, and Christa Schleper. Novel genes for nitrite reductase and Amo-related proteins indicate a role of uncultivated mesophilic crenarchaeota in nitrogen cycling. *Environmental microbiology*, 7(12):1985–1995, December 2005.

[150] Susannah Green Tringe, Christian von Mering, Arthur Kobayashi, Asaf A Salamov, Kevin Chen, Hwai W Chang, Mircea Podar, Jay M Short, Eric J Mathur, J Chris Detter, Peer Bork, Philip Hugenholtz, and Edward M Rubin. Comparative metagenomics of microbial communities. *Science (New York, NY)*, 308(5721):554–557, April 2005.

[151] Peter J Turnbaugh, Micah Hamady, Tanya Yatsunenko, Brandi L Cantarel, Alexis Duncan, Ruth E Ley, Mitchell L Sogin, William J Jones, Bruce A Roe, Jason P Affourtit, Michael Egholm, Bernard Henrissat, Andrew C Heath, Rob Knight, and Jeffrey I Gordon. A core gut microbiome in obese and lean twins. *Nature*, 457(7228):480–484, January 2009.

[152] Michael Tushman and Philip Anderson. Technological Discontinuities and Organizational Environments. *Administrative Science Quarterly*, 31(3):439–465, September 1986.

[153] Gene W Tyson, Jarrod Chapman, Philip Hugenholtz, Eric E Allen, Rachna J Ram, Paul M Richardson, Victor V Solovyev, Edward M Rubin, Daniel S Rokhsar, and Jillian F Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 428(6978):37–43, March 2004.

[154] J C Venter, M D Adams, E W Myers, P W Li, R J Mural, G G Sutton, H O Smith, M Yandell, C A Evans, R A Holt, J D Gocayne, P Amanatides, R M Ballew, D H Hu-

son, J R Wortman, Q Zhang, C D Kodira, XQH Zheng, L Chen, M Skupski, G Subramanian, P D Thomas, J H Zhang, GLG Miklos, C Nelson, S Broder, A G Clark, C Nadeau, V A McKusick, N Zinder, A J Levine, R J Roberts, M Simon, C Slayman, M Hunkapiller, R Bolanos, A Delcher, I Dew, D Fasulo, M Flanigan, L Florea, A Halpern, S Hannenhalli, S Kravitz, S Levy, C Mobarry, K Reinert, K Remington, J Abu-Threideh, E Beasley, K Biddick, V Bonazzi, R Brandon, M Cargill, I Chandramouliswaran, R Charlab, K Chaturvedi, Z M Deng, V Di Francesco, P Dunn, K Eilbeck, C Evangelista, A E Gabrielian, W Gan, W M Ge, F C Gong, Z P Gu, P Guan, T J Heiman, M E Higgins, R R Ji, Z X Ke, K A Ketchum, Z W Lai, Y D Lei, Z Y Li, J Y Li, Y Liang, X Y Lin, F Lu, G V Merkulov, N Milshina, H M Moore, A K Naik, V A Narayan, B Neelam, D Nusskern, D B Rusch, S Salzberg, W Shao, B X Shue, J T Sun, Z Y Wang, A H Wang, X Wang, J Wang, M H Wei, R Wides, C L Xiao, C H Yan, A Yao, J Ye, M Zhan, W Q Zhang, H Y Zhang, Q Zhao, L S Zheng, F Zhong, W Y Zhong, SPC Zhu, S Y Zhao, D Gilbert, S Baumhueter, G Spier, C Carter, A Cravchik, T Woodage, F Ali, H J An, A Awe, D Baldwin, H Baden, M Barnstead, I Barrow, K Beeson, D Busam, A Carver, Center, A, M L Cheng, L Curry, S Danaher, L Davenport, R Desilets, S Dietz, K Dodson, L Doup, S Ferriera, N Garg, A Gluecksmann, B Hart, J Haynes, C Haynes, C Heiner, S Hladun, D Hostin, J Houck, T Howland, C Ibegwam, J Johnson, F Kalush, L Kline, S Koduru, A Love, F Mann, D May, S McCawley, T McIntosh, I McMullen, M Moy, L Moy, B Murphy, K Nelson, C Pfannkoch, E Pratts, V Puri, H Qureshi, M Reardon, R Rodriguez, Y H Rogers, D Romblad, B Ruhfel, R Scott, C Sitter, M Smallwood, E Stewart, R Strong, E Suh, R Thomas, N N Tint, S Tse, C Vech, G Wang, J Wetter, S Williams, M Williams, S Windsor, E Winn-Deen, K Wolfe, J Zaveri, K Zaveri, J F Abril, R Guigo, M J Campbell, K V Sjolander, B Karlak, A Kejariwal, H Y Mi, B Lazareva, T Hatton, A Narechania, K Diemer, A Muruganujan, N Guo, S Sato, V Bafna, S Istrail, R Lippert, R Schwartz, B Walenz, S Yooseph, D Allen, A Basu, J Baxendale, L Blick, M Caminha, J Carnes-Stine, P Caulk, Y H Chiang, M Coyne, C Dahlke, A D Mays, M Dombroski, M Donnelly, D Ely, S Esparham, C Fosler, H Gire, S Glanowski, K Glasser, A Glodek, M Gorokhov, K Graham, B Gropman, M Harris, J Heil, S Henderson, J Hoover, D Jennings, C Jordan, J Jordan, J Kasha, L Kagan, C Kraft, A Levitsky, M Lewis, X J Liu, J Lopez, D Ma, W Majoros, J McDaniel, S Murphy, M Newman, T Nguyen, N Nguyen, M Nodell, S Pan, J Peck, M Peterson, W Rowe, R Sanders, J Scott, M Simpson, T Smith, A Sprague, T Stockwell, R Turner, E Venter, M Wang, M Y Wen, D Wu, M Wu, A Xia, A Zandieh, and X H Zhu. The Sequence of the Human Genome. *Science (New York, NY)*, 291(5507):1304–1351, February 2001.

[155] J Craig Venter, Karin Remington, John F Heidelberg, Aaron L Halpern, Douglas B Rusch, Jonathan A Eisen, Dongying Wu, Ian Paulsen, Karen E Nelson, William C

Nelson, Derrick E Fouts, Samuel Levy, Anthony H Knap, Michael W Lomas, Kenneth Nealson, Owen White, Jeremy Peterson, Jeff M Hoffman, Rachel Parsons, Holly Baden-Tillson, Cynthia Pfannkoch, Yu-Hui Rogers, and Hamilton O Smith. Environmental genome shotgun sequencing of the Sargasso Sea. *Science (New York, NY)*, 304(5667):66–74, April 2004.

[156] C B Walker, J R de la Torre, M G Klotz, H Urakawa, N Pinel, D J Arp, C Brochier-Armanet, P S G Chain, P P Chan, A Gollabgir, J Hemp, M Hügler, E A Karr, M Könneke, M Shin, T J Lawton, T M Lowe, W Martens-Habbena, L Sayavedra-Soto, D Lang, S M Sievert, A C Rosenzweig, G Manning, and D A Stahl. Nitrosopumilus maritimus genome reveals unique mechanisms for nitrification and autotrophy in globally distributed marine crenarchaea. *Proceedings of the National Academy of Sciences of the United States of America*, 107(19):8818–8823, May 2010.

[157] David A Walsh, Elena Zaikova, Charles G Howes, Young C Song, Jody J Wright, Susannah Green Tringe, Philippe D Tortell, and Steven J Hallam. Metagenome of a Versatile Chemolithoautotroph from Expanding Oceanic Dead Zones. *Science (New York, NY)*, 326(5952):578–582, 2009.

[158] Jessica M Walter, Derek Greenfield, Carlos Bustamante, and Jan Liphardt. Light-powering Escherichia coli with proteorhodopsin. *Proceedings of the National Academy of Sciences of the United States of America*, 104(7):2408–2412, February 2007.

[159] Qiong Wang, George M Garrity, James M Tiedje, and James R Cole. Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied And Environmental Microbiology*, 73(16):5261–5267, August 2007.

[160] Falk Warnecke and Philip Hugenholtz. Building on basic metagenomics with complementary technologies. *Genome biology*, 2007.

[161] Markus G Weinbauer. Ecology of prokaryotic viruses. *Fems Microbiology Reviews*, 28(2):127–181, May 2004.

[162] William B Whitman, David C Coleman, and William J Wiebe. Prokaryotes: the unseen majority. *Proceedings of the National Academy of Sciences of the United States of America*, 95(12):6578–6583, June 1998.

[163] Shannon J Williamson, Douglas B Rusch, Shibu Yooseph, Aaron L Halpern, Karla B Heidelberg, John I Glass, Cynthia Andrews-Pfannkoch, Douglas Fadrosh, Christopher S Miller, Granger Sutton, Marvin Frazier, and J Craig Venter. The Sorcerer II Global Ocean Sampling Expedition: metagenomic characterization of viruses within aquatic microbial samples. *Plos One*, 3(1):e1456, 2008.

[164] Carl R Woese and George E Fox. Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proceedings of the National Academy of Sciences of the United States of America*, 74(11):5088–5090, November 1977.

[165] K Eric Wommack and Rita R Colwell. Virioplankton: Viruses in Aquatic Ecosystems. *Microbiology and Molecular Biology Reviews*, 64(1):69–114, March 2000.

[166] Tanja Woyke, Gary Xie, Alex Copeland, José M González, Cliff S Han, Hajnalka Kiss, Jimmy H Saw, Pavel Senin, Chi Yang, Sourav Chatterji, Jan-Fang Cheng, Jonathan A Eisen, Michael E Sieracki, and Ramunas Stepanauskas. Assembling the marine metagenome, one cell at a time. *Plos One*, 4(4):e5299, 2009.

[167] Shibu Yooseph, Granger Sutton, Douglas B Rusch, Aaron L Halpern, Shannon J Williamson, Karin Remington, Jonathan A Eisen, Karla B Heidelberg, Gerard Manning, Weizhong Li, Lukasz Jaroszewski, Piotr Cieplak, Christopher S Miller, Huiying Li, Susan T Mashiyama, Marcin P Joachimiak, Christopher van Belle, John-Marc Chandonia, David A Soergel, Yufeng Zhai, Kannan Natarajan, Shaun Lee, Benjamin J Raphael, Vineet Bafna, Robert Friedman, Steven E Brenner, Adam Godzik, David Eisenberg, Jack E Dixon, Susan S Taylor, Robert L Strausberg, Marvin Frazier, and J Craig Venter. The Sorcerer II Global Ocean Sampling expedition: expanding the universe of protein families. *PLoS biology*, 5(3):e16, March 2007.

[168] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, May 2008.

[169] Yanlin Zhao, Ben Temperton, J Cameron Thrash, Michael S Schwalbach, Kevin L Vergin, Zachary C Landry, Mark Ellisman, Tom Deerinck, Matthew B Sullivan, and Stephen J Giovannoni. Abundant SAR11 viruses in the ocean. *Nature*, 494(7437):357–360, February 2013.

Appendix A

SUPPLEMENTAL FIGURES

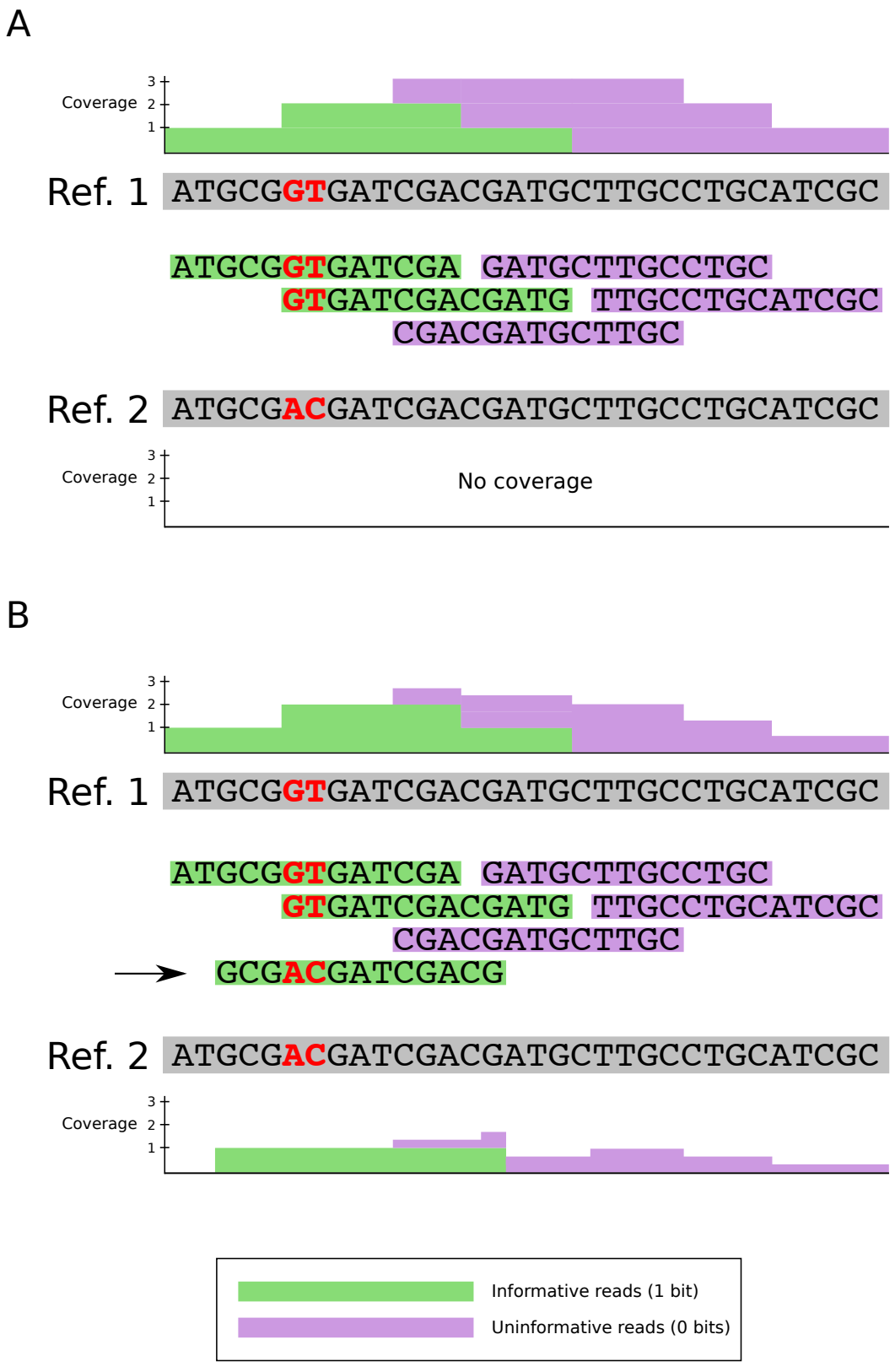
A
Coverage
3
2
1
Ref. 1
ATGCGGTGATCGACGATGCTTGCCTGCATCGC
ATGCGGTGATCGA
GATGCTTGCCTGC
GTGATCGACGATG
TTGCCTGCATCGC
CGACGATGCTTGC
Ref. 2
ATGCGACGATCGACGATGCTTGCCTGCATCGC
Coverage
3
2
1
No coverage
B
Coverage
3
2
1
Ref. 1
ATGCGGTGATCGACGATGCTTGCCTGCATCGC
ATGCGGTGATCGA
GATGCTTGCCTGC
GTGATCGACGATG
TTGCCTGCATCGC
CGACGATGCTTGC
GCGACGATCGACG
Ref. 2
ATGCGACGATCGACGATGCTTGCCTGCATCGC
Coverage
3
2
1
Informative reads (1 bit)
Uninformative reads (0 bits)

Figure A.1: Illustration of reference selection, fractional coverage allocation, and relative abundance estimation using sequence read alignments to a reference database. The examples in this figure use a simple database with only two reference sequences ("Ref. 1" and "Ref. 2"). These two sequences differ by only two nucleotides (red). Aligned reads are shown located between the two references, and are shaded by information content relative to the reference database. Informative reads (green) each provide 1 bit of information, because they align equally well with one of the two references ($-\log_2 ({}^1/_2) = 1$ bit). Uninformative reads (purple) each provide 0 bits of information, because they align equally well with both of the references ($-\log_2 ({}^2/_2) = 0$ bits).

(A) In this example, Ref. 1 is selected from the database because reads scoring 2 bits have best alignments with it. Ref. 2 is not selected because no informative reads have a best alignment with it. All coverage from aligned reads (shown on graphs) goes to Ref. 1 because it is the only selected reference. Its relative abundance is 100% because its mean coverage equals the total coverage from all reads.

(B) This example is identical to (A) above, except that an additional informative read (arrow) is added with a best alignment to Ref. 2. Ref. 1 is selected from the database first because 2 bits of reads best align with it, and that is the maximum for all reference sequences. Ref. 2 is selected next from the database because it has 1 bit of (residual) information after reads that have best alignments with previously selected references are removed from consideration (e.g. if Ref 1. and Ref 2. were identical, only one of them could be selected). Coverage for reads with unique best alignments is allocated first (shown on graphs in green), and coverage for reads aligning equivalently with the two references is allocated fractionally (i.e. shared) with weighting determined by the relative mean amount of previously accumulated coverage (in this case ${}^2/_3$ for Ref. 1 and ${}^1/_3$ for Ref. 2, shown on graphs in purple). Once the coverage for all reads is allocated, the relative abundance for the selected sequences is calculated from the mean coverage values relative to total coverage allocated: Ref. 1 ~66.7%, Ref. 2 ~33.3%.

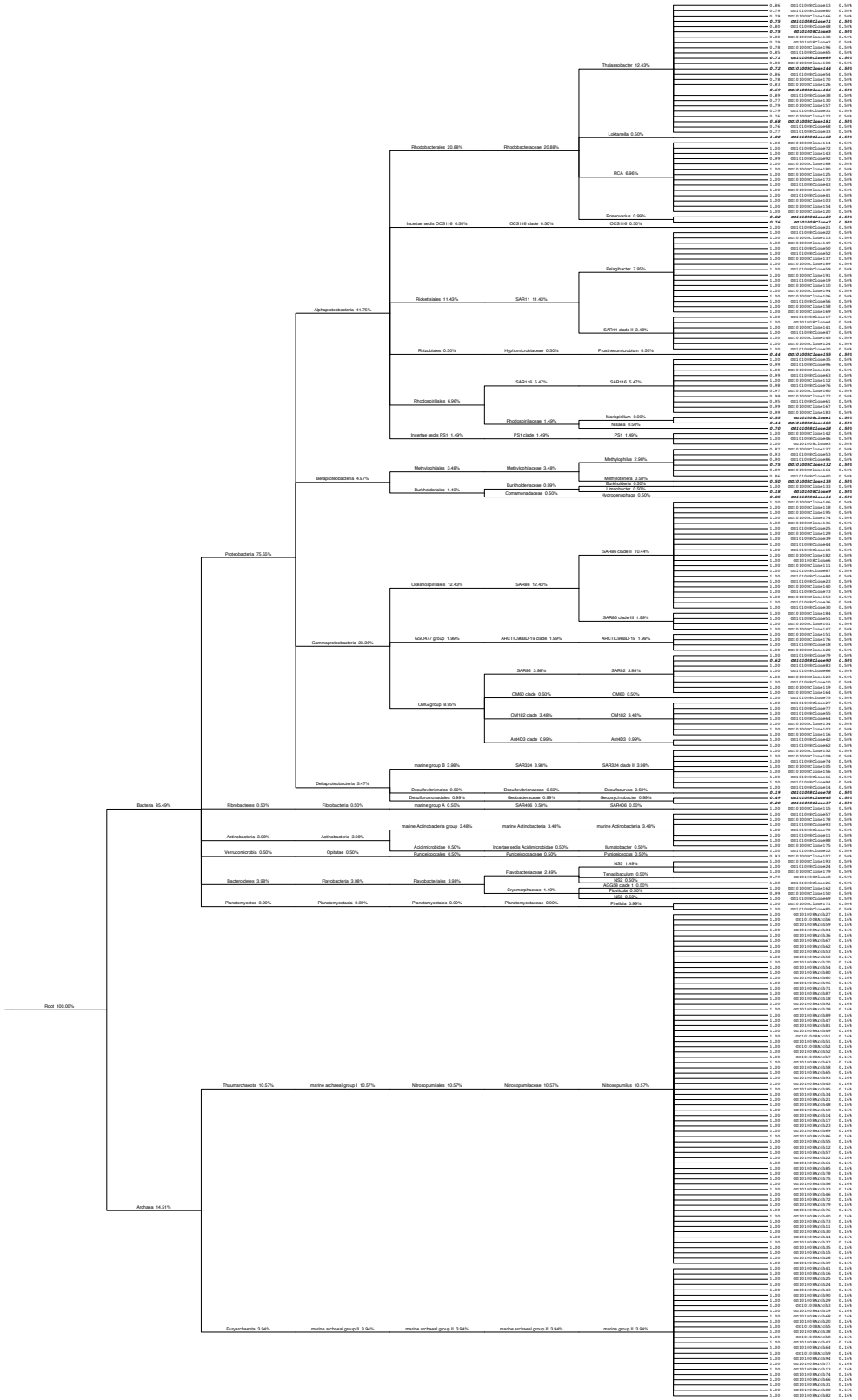
Root 100.00%
Bacteria 85.49%
Archaea 14.51%
Fibrobacteres 0.50%
Actinobacteria 3.98%
Verrucomicrobia 0.50%
Bacteroidetes 3.98%
Planctomycetes 0.99%
Thaumarchaeota 10.57%
Euryarchaeota 3.94%
Proteobacteria 75.55%
Fibrobacteria 0.50%
Actinobacteria 3.98%
Opitutae 0.50%
Flavobacteria 2.98%
Planctomycetacia 0.99%
marine archaeal group I 10.57%
marine archaeal group II 3.94%
Alphaproteobacteria 41.75%
Betaproteobacteria 4.97%
Gammaproteobacteria 23.36%
Deltaproteobacteria 5.47%
Rhodobacterales 20.88%
Rickettsiales 11.43%
Rhizobiales 0.50%
Rhodospirillales 6.96%
Incertae sedis PS1 1.49%
Methylophilales 3.48%
Burkholderiales 1.49%
Oceanospirillales 12.43%
GSO477 group 1.99%
OM1 group 8.95%
marine group B 3.98%
Desulfovibrionales 0.50%
Desulfuromonadales 0.99%
marine group A 0.50%
marine Actinobacteria group 3.48%
Acidimicrobiidae 0.50%
Pyrinomonadales 0.99%
Flavobacteriales 2.98%
Planctomycetales 0.99%
Nitrosopumilales 10.57%
marine archaeal group II 3.94%
Rhodobacteraceae 20.88%
SAR11 11.43%
Hyphomicrobiaceae 0.50%
Rhodospirillaceae 1.49%
PS1 clade 1.49%
Methylophilaceae 3.48%
Burkholderiaceae 0.99%
Comamonadaceae 0.50%
SAR86 12.43%
ARCTIC96BD-19 clade 1.99%
SAR92 3.98%
OM60 clade 0.50%
OM182 clade 3.48%
SAR324 3.98%
Desulfovibrionaceae 0.50%
Geobacteraceae 0.99%
SAR406 0.50%
marine Actinobacteria 3.48%
Incertae sedis Acidimicrobidae 0.50%
Pyrinomonadaceae 0.50%
Flavobacteriaceae 2.49%
Cryomorphaceae 1.49%
Planctomycetaceae 0.99%
Nitrosopumilaceae 10.57%
marine archaeal group II 3.94%
Thalassobacter 12.43%
Loktanella 0.50%
RCA 6.96%
Roseovarius 0.99%
OCS116 clade 0.50%
Incertae sedis OCS116 0.50%
Pelagibacter 7.95%
SAR11 clade II 3.48%
Psuetheomicrobium 0.50%
SAR116 5.47%
Marispirillum 0.99%
Nocea 0.50%
PS1 1.49%
Methylophilus 2.98%
Methylotenera 0.50%
Limnobacter 0.50%
Hydrogenophaga 0.50%
SAR86 clade II 10.44%
SAR86 clade III 1.99%
ARCTIC96BD-19 1.99%
SAR92 3.98%
OM60 0.50%
OM182 3.48%
Arn4D3 0.99%
SAR324 clade II 3.98%
Geopsychrobacter 0.99%
SAR406 0.50%
Ilumatobacter 0.50%
Pyrinomonas 0.50%
NS5 1.49%
Tenacibaculum 0.50%
NS2 0.99%
Arenibacter 0.99%
Flavicella 0.50%
NS9 0.50%
Fluviicola 0.99%
Nitrosopumilus 10.57%
marine group II 3.94%

Figure A.2: Taxonomic cladogram showing the Bayesian classification of all 16S rDNA clones generated from the October 2008 sample. Taxonomic groups, labeled on branches, are from the RDP taxonomy extended with marine environmental clades (see Methods). Clone identifiers are noted at the leaves. Bayesian classification p-values for the clone sequences are shown to the left of the clone identifiers. Percentages noted on each branch and leaf represent the abundance of that taxonomic unit as a fraction of total clones generated from the October 2008 sample, with the fraction of bacterial and archaeal clones fixed as 85.49% and 14.51% of the population respectively, as determined from aligning metagenomic reads to the October 16S clone sequences (Fig. A.8). Clones identified in bold-italic type were excluded from abundance analysis simulations.

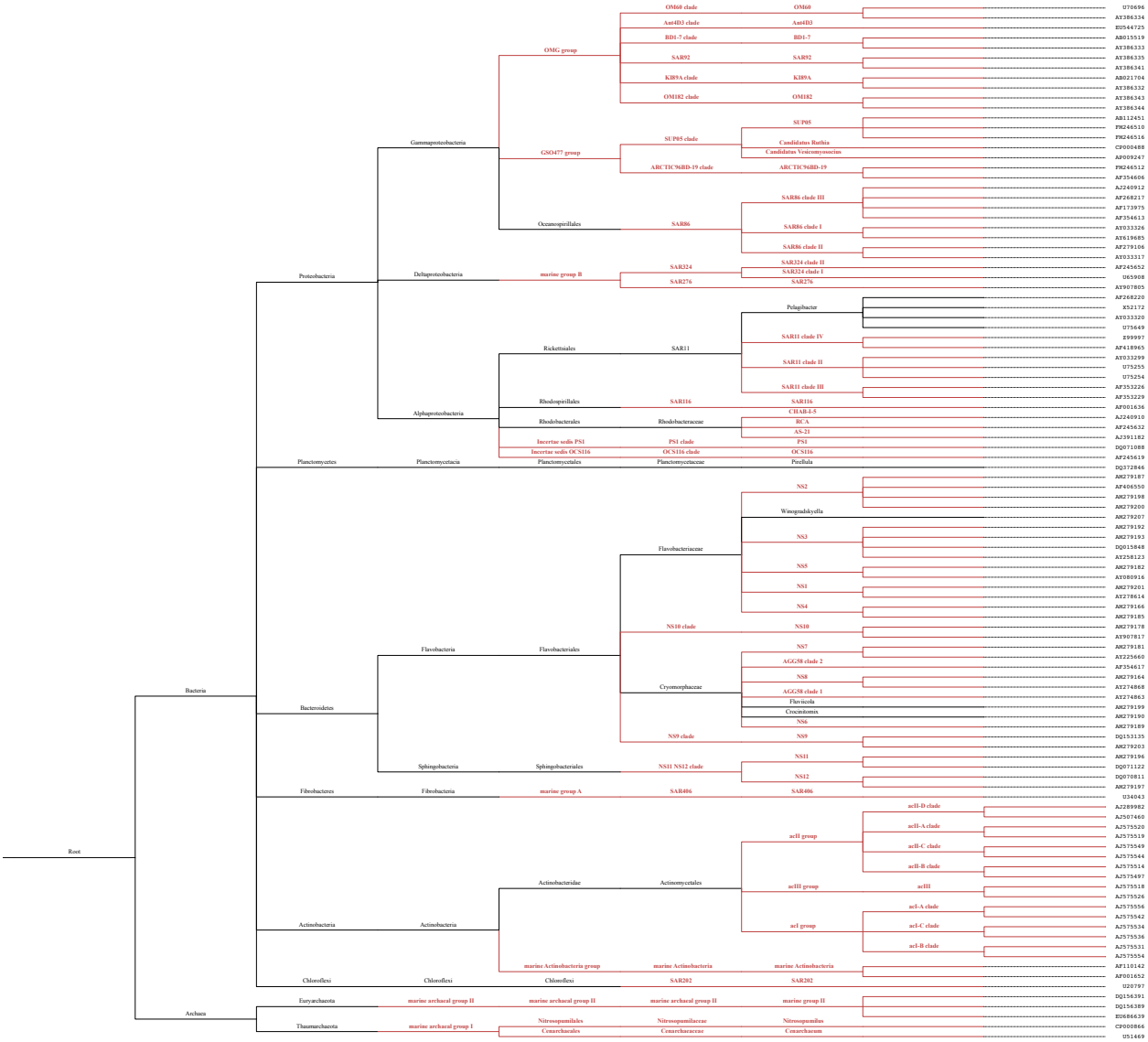

Figure A.3: Taxonomic cladogram of environmental 16S rDNA sequences added to the training set of the RDP Bayesian classifier. Branches in black denote pre-existing RDP taxonomy. Red branches denote environmental clades added to the RDP taxonomy for this study. Leaves of the tree show accession numbers of the sequences added to the training set.

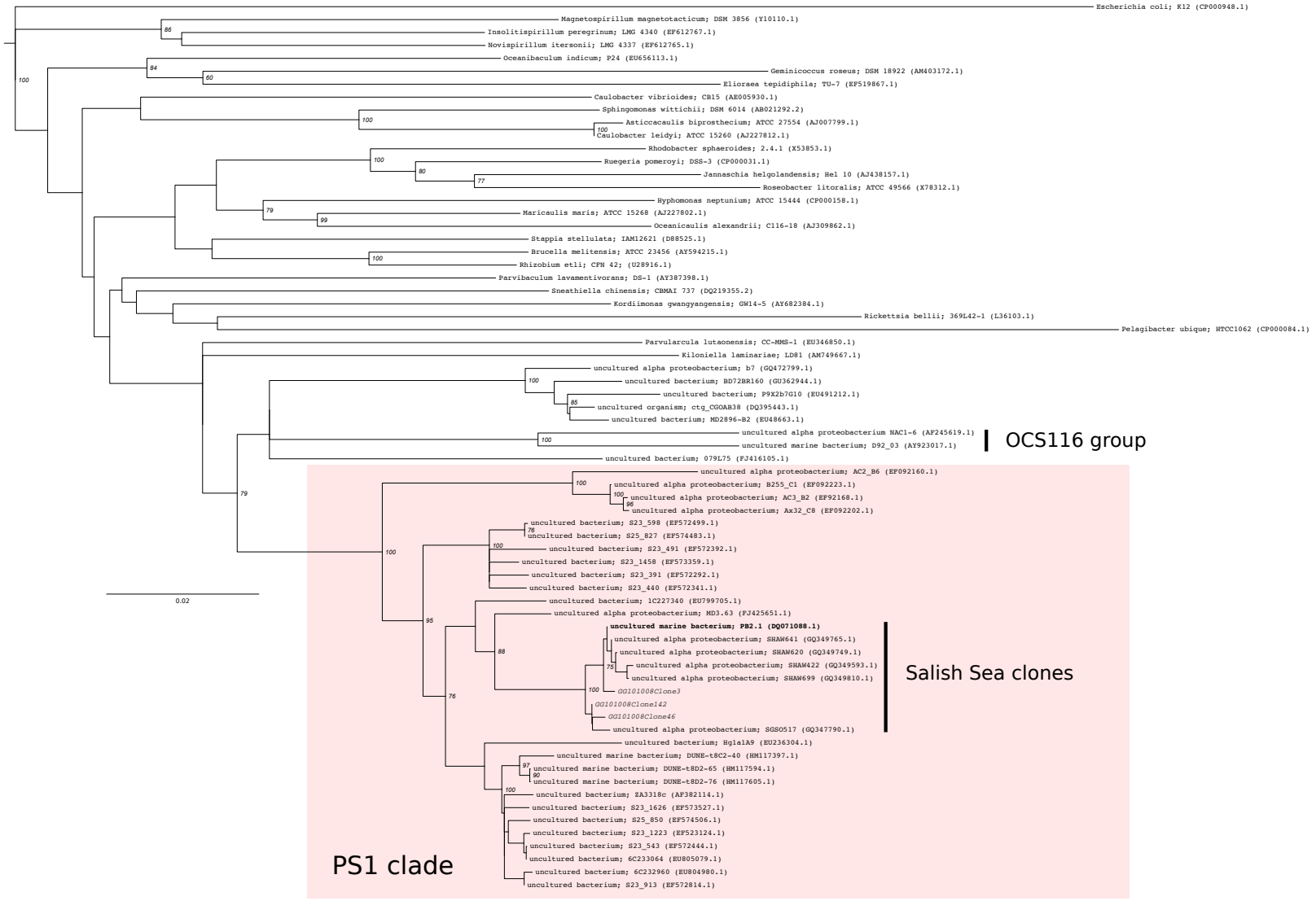

Figure A.4: Neighbor-joining phylogeny of alpha-*Proteobacterial* 16S rDNA sequences, showing a novel clade of marine alpha-*Proteobacteria* designated as “PS1”. The tree is rooted with *Escherichia coli*, a member of the gamma-*Proteobacteria*. Branch lengths are proportional to the number of substitutions per site. Bootstraps are shown for 100 replicate trees. The PS1 clade is boxed in pink. A sub-clade of sequences isolated from the vicinity of Puget Sound, WA, USA and the Straights of Georgia, BC, Canada are noted as “Salish Sea clones”. The taxon in bold type was sequenced from Puget Sound in a previous study. Taxa in italic type are clones sequenced in this study.

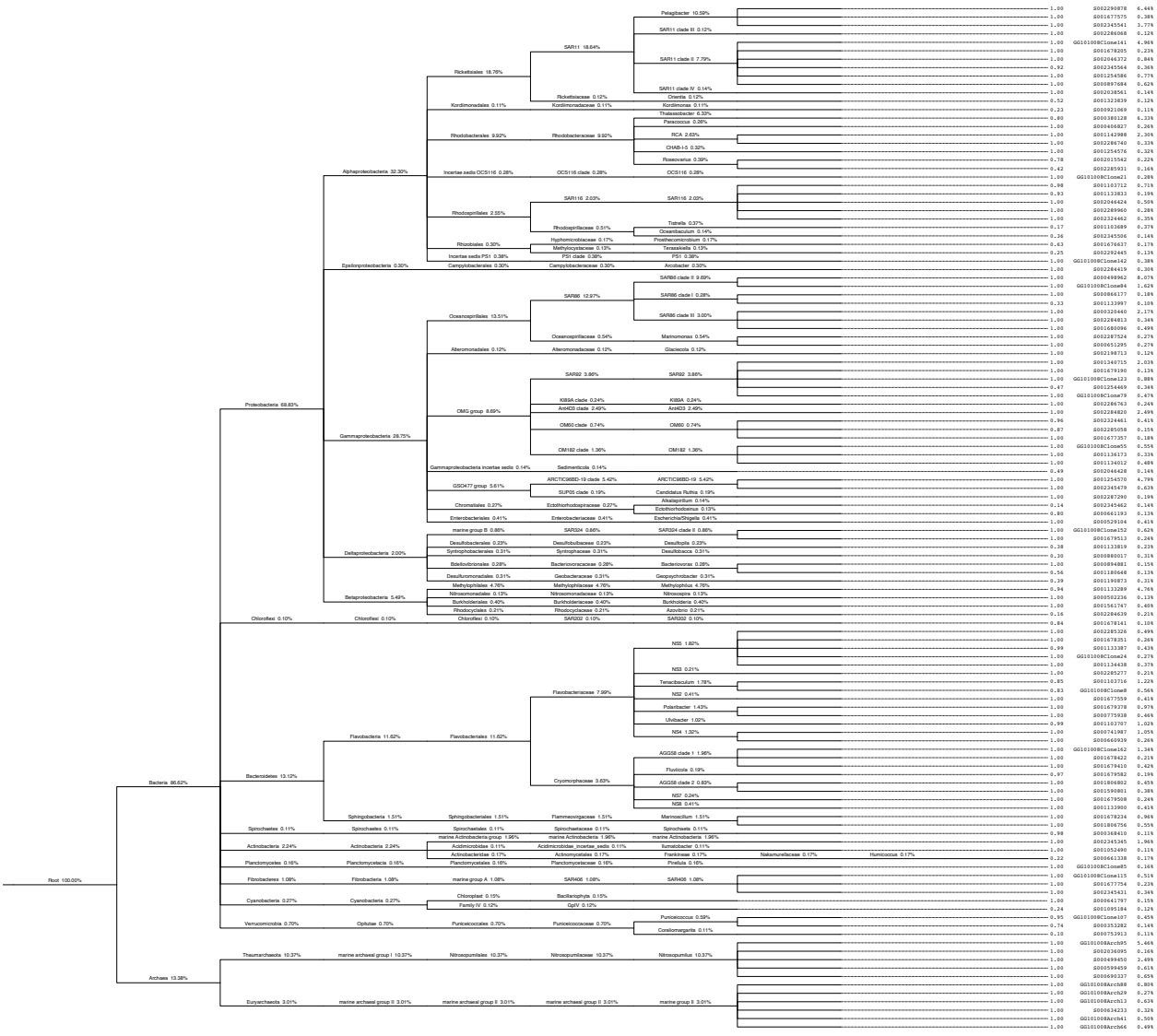

Figure A.5: Taxonomic cladogram detailing all 16S rDNA sequences selected for the October 2008 sample. Taxonomic groups, labeled on branches, are from the extended RDP taxonomy. RDP identifiers for selected sequences are noted at the leaves, except for those prefaced with "GG" which denote clones generated in this study. Bayesian classification p-values for the selected sequences are shown to the left of the clone identifiers. Percentages noted on each branch and leaf represent the estimated abundance of that taxonomic unit as a fraction of total 16S rDNA in the sample.

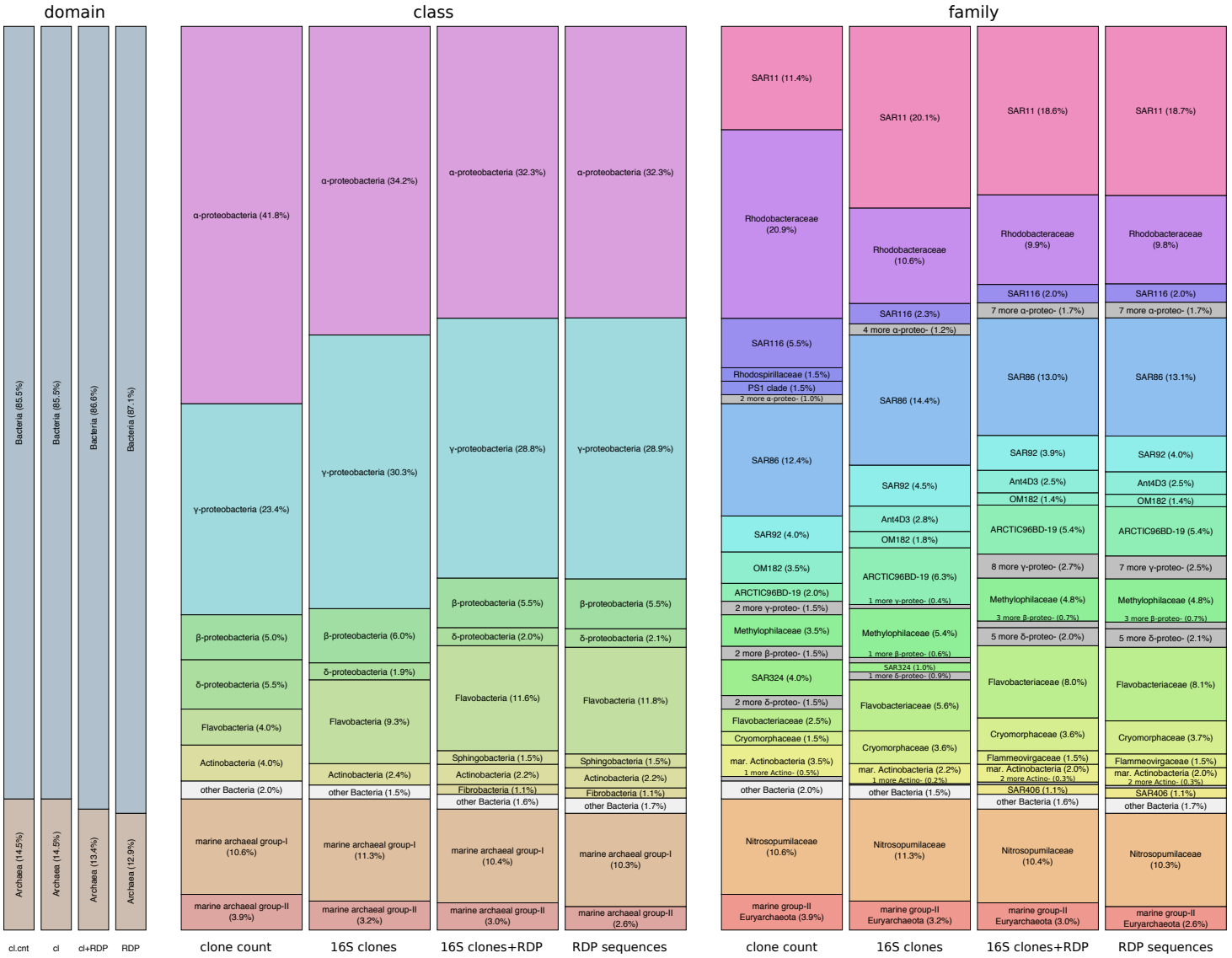
domain
class
family
Bacteria (85.5%)
Bacteria (85.5%)
Bacteria (86.6%)
Bacteria (87.1%)
Archaea (14.5%)
Archaea (14.5%)
Archaea (13.4%)
Archaea (12.9%)
α-proteobacteria (41.8%)
γ-proteobacteria (23.4%)
β-proteobacteria (5.0%)
δ-proteobacteria (5.5%)
Flavobacteria (4.0%)
Actinobacteria (4.0%)
other Bacteria (2.0%)
marine archaeal group-I (10.6%)
marine archaeal group-II (3.9%)
α-proteobacteria (34.2%)
γ-proteobacteria (30.3%)
β-proteobacteria (6.0%)
δ-proteobacteria (1.9%)
Flavobacteria (9.3%)
Actinobacteria (2.4%)
other Bacteria (1.5%)
marine archaeal group-I (11.3%)
marine archaeal group-II (3.2%)
α-proteobacteria (32.3%)
γ-proteobacteria (28.8%)
β-proteobacteria (5.5%)
δ-proteobacteria (2.0%)
Flavobacteria (11.6%)
Sphingobacteria (1.5%)
Actinobacteria (2.2%)
Fibrobacteria (1.1%)
other Bacteria (1.6%)
marine archaeal group-I (10.4%)
marine archaeal group-II (3.0%)
α-proteobacteria (32.3%)
γ-proteobacteria (28.9%)
β-proteobacteria (5.5%)
δ-proteobacteria (2.1%)
Flavobacteria (11.8%)
Sphingobacteria (1.5%)
Actinobacteria (2.2%)
Fibrobacteria (1.1%)
other Bacteria (1.7%)
marine archaeal group-I (10.3%)
marine archaeal group-II (2.6%)
SAR11 (11.4%)
Rhodobacteraceae (20.9%)
SAR116 (5.5%)
Rhodospirillaceae (1.5%)
PS1 clade (1.5%)
2 more α-proteo- (1.0%)
SAR86 (12.4%)
SAR92 (4.0%)
OM182 (3.5%)
ARCTIC96BD-19 (2.0%)
2 more γ-proteo- (1.5%)
Methylophilaceae (3.5%)
2 more β-proteo- (1.5%)
SAR324 (4.0%)
2 more δ-proteo- (1.5%)
Flavobacteriaceae (2.5%)
Cryomorphaceae (1.5%)
mar. Actinobacteria (3.5%)
1 more Actino- (0.5%)
other Bacteria (2.0%)
Nitrosopumilaceae (10.6%)
marine group-II Euryarchaeota (3.9%)
SAR11 (20.1%)
Rhodobacteraceae (10.6%)
SAR116 (2.3%)
4 more α-proteo- (1.2%)
SAR86 (14.4%)
SAR92 (4.5%)
Ant4D3 (2.8%)
OM182 (1.8%)
ARCTIC96BD-19 (6.3%)
1 more γ-proteo- (0.4%)
Methylophilaceae (5.4%)
1 more β-proteo- (0.6%)
SAR324 (1.0%)
1 more δ-proteo- (0.9%)
Flavobacteriaceae (5.6%)
Cryomorphaceae (3.6%)
mar. Actinobacteria (2.2%)
1 more Actino- (0.2%)
other Bacteria (1.5%)
Nitrosopumilaceae (11.3%)
marine group-II Euryarchaeota (3.2%)
SAR11 (18.6%)
Rhodobacteraceae (9.9%)
SAR116 (2.0%)
7 more α-proteo- (1.7%)
SAR86 (13.0%)
SAR92 (3.9%)
Ant4D3 (2.5%)
OM182 (1.4%)
ARCTIC96BD-19 (5.4%)
8 more γ-proteo- (2.7%)
Methylophilaceae (4.8%)
3 more β-proteo- (0.7%)
5 more δ-proteo- (2.0%)
Flavobacteriaceae (8.0%)
Cryomorphaceae (3.6%)
Flammeovirgaceae (1.5%)
mar. Actinobacteria (2.0%)
2 more Actino- (0.3%)
SAR406 (1.1%)
other Bacteria (1.6%)
Nitrosopumilaceae (10.4%)
marine group-II Euryarchaeota (3.0%)
SAR11 (18.7%)
Rhodobacteraceae (9.8%)
SAR116 (2.0%)
7 more α-proteo- (1.7%)
SAR86 (13.1%)
SAR92 (4.0%)
Ant4D3 (2.5%)
OM182 (1.4%)
ARCTIC96BD-19 (5.4%)
7 more γ-proteo- (2.5%)
Methylophilaceae (4.8%)
3 more β-proteo- (0.7%)
5 more δ-proteo- (2.1%)
Flavobacteriaceae (8.1%)
Cryomorphaceae (3.7%)
Flammeovirgaceae (1.5%)
mar. Actinobacteria (2.0%)
2 more Actino- (0.3%)
SAR406 (1.1%)
other Bacteria (1.7%)
Nitrosopumilaceae (10.3%)
marine group-II Euryarchaeota (2.6%)
cl.cnt
cl
cl+RDP
RDP
clone count
16S clones
16S clones+RDP
RDP sequences
clone count
16S clones
16S clones+RDP
RDP sequences

Figure A.6: Relative abundance of Bacterial and Archaeal taxonomic groups for the October sample, a comparison of four different estimation methods. Stacked bars correspond to the relative prokaryotic 16S abundance for hierarchical groups at the domain, class and family taxonomic levels (or the approximate equivalent for environmental clades). The abundance estimation methods are: October 2008 16S rDNA clone counts (cl.cnt) as shown in Figure A.2; and metagenomic read alignments to October 16S clones only (cl), October 16S clones added to the RDP 16S database (cl+RDP) and the RDP 16S database only (RDP). Colors correspond to the same groups for each estimation method. White bars contain classes (and their families) that are < 1% relative abundance. Grey bars contain families within a class that are < 1%.

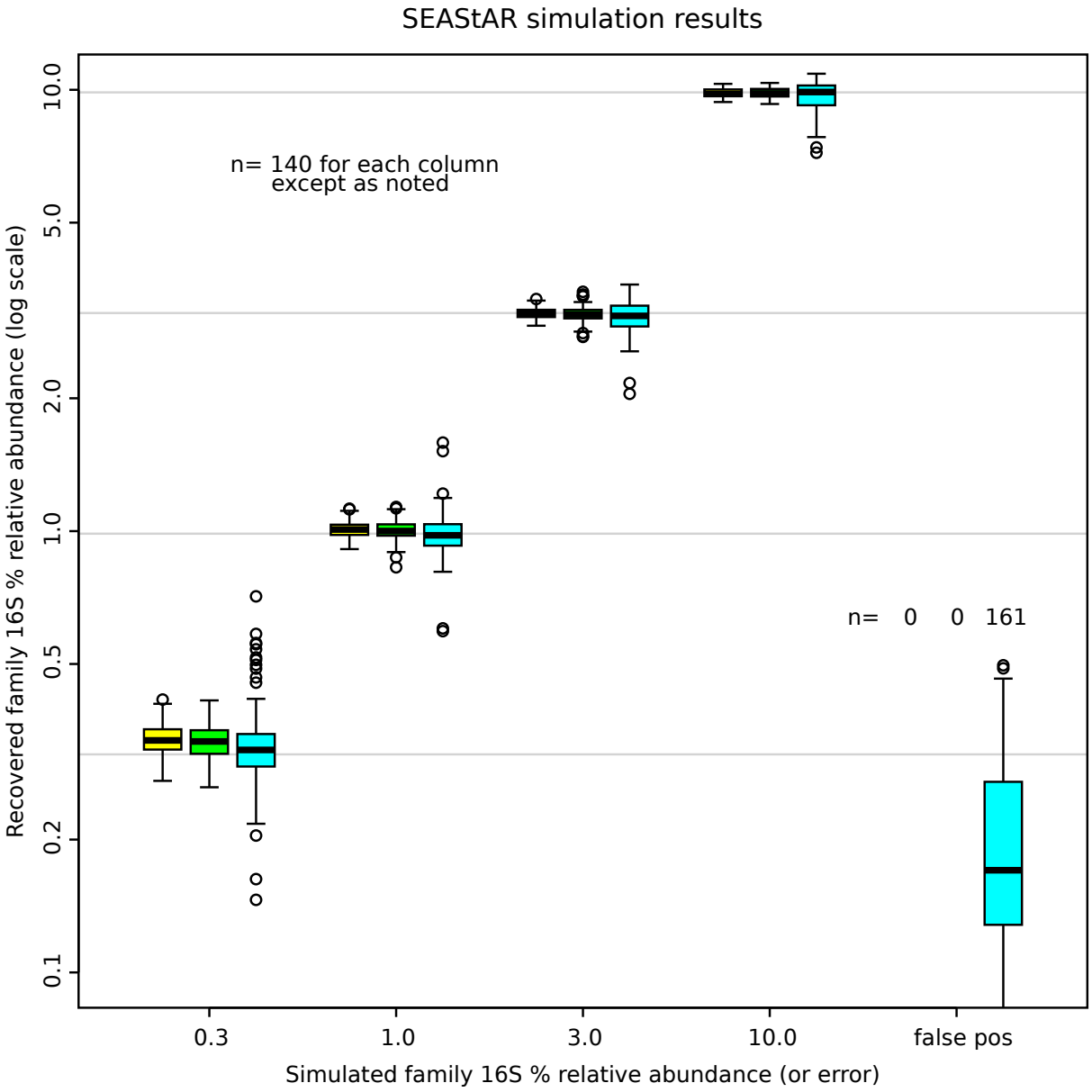
SEAStAR simulation results
n= 140 for each column
except as noted
n= 0 0 161
Recovered family 16S % relative abundance (log scale)
10.0
5.0
2.0
1.0
0.5
0.2
0.1
0.3
1.0
3.0
10.0
false pos
Simulated family 16S % relative abundance (or error)

Figure A.7: Results of abundance estimation for synthetic metagenomic datasets generated from simulated populations of 16S rDNA sequences. This figure depicts simulation results from the analysis of twenty synthetic metagenomic datasets, each constructed by randomly drawing a fixed number of reads from significantly larger pools generated from randomly selected 16S sequences in fixed proportions, shown on the y-axis. The x-axis shows the four classes of simulated taxa abundances that were successfully recovered and classified to the family taxonomic level. A fifth category (false pos) on the x-axis contains taxa that were detected, but which did not correspond (at the family level) to one of the taxa used to generate that simulated population. There were no false negatives detected (i.e. all taxa in all trial populations were recovered in the simulations). The three columns within each x-axis category are for analysis using different reference databases (left to right): October 16S clones only (yellow), October 16S clones added to the RDP 16S database (green), and the RDP 16S database only (blue). The y-axis shows the estimated abundance of recovered family-level taxa on a log scale. The box-and-whiskers plotted show the median value (black bar), the interquartile (filled box), range of values (whiskers), and outliers beyond 1.5-times the range of the interquartile (open circles). Grey horizontal lines show the simulated initial populations of the four abundance classes.

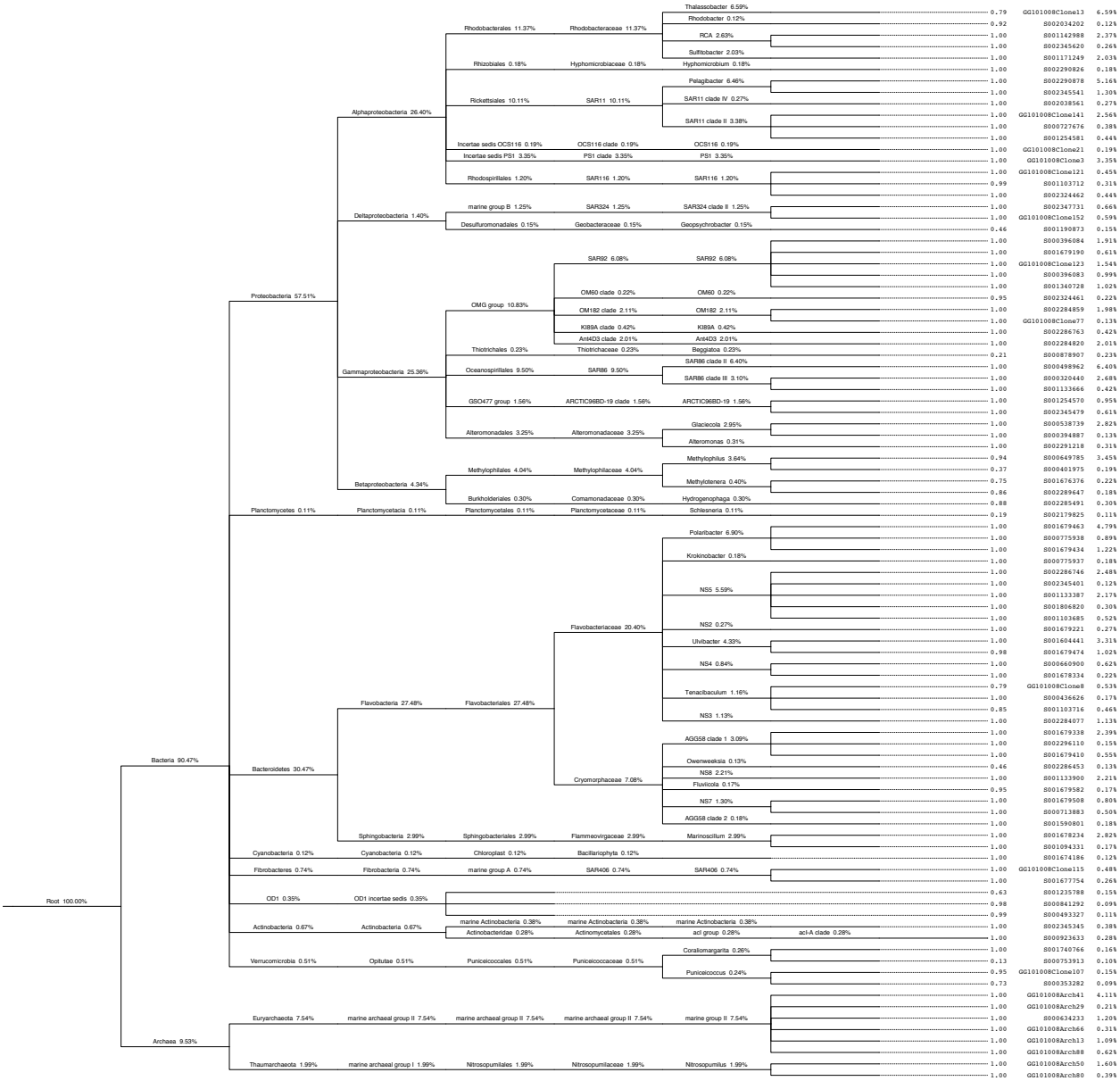

Figure A.8: Taxonomic cladogram detailing all 16S rDNA sequences selected for the May 2009 sample. Taxonomic groups, labeled on branches, are from the extended RDP taxonomy. RDP identifiers for selected sequences are noted at the leaves, except for those prefaced with “GG” which denote clones generated in this study. Bayesian classification p-values for the selected sequences are shown to the left of the clone identifiers. Percentages noted on each branch and leaf represent the estimated abundance of that taxonomic unit as a fraction of total 16S rDNA in the sample.

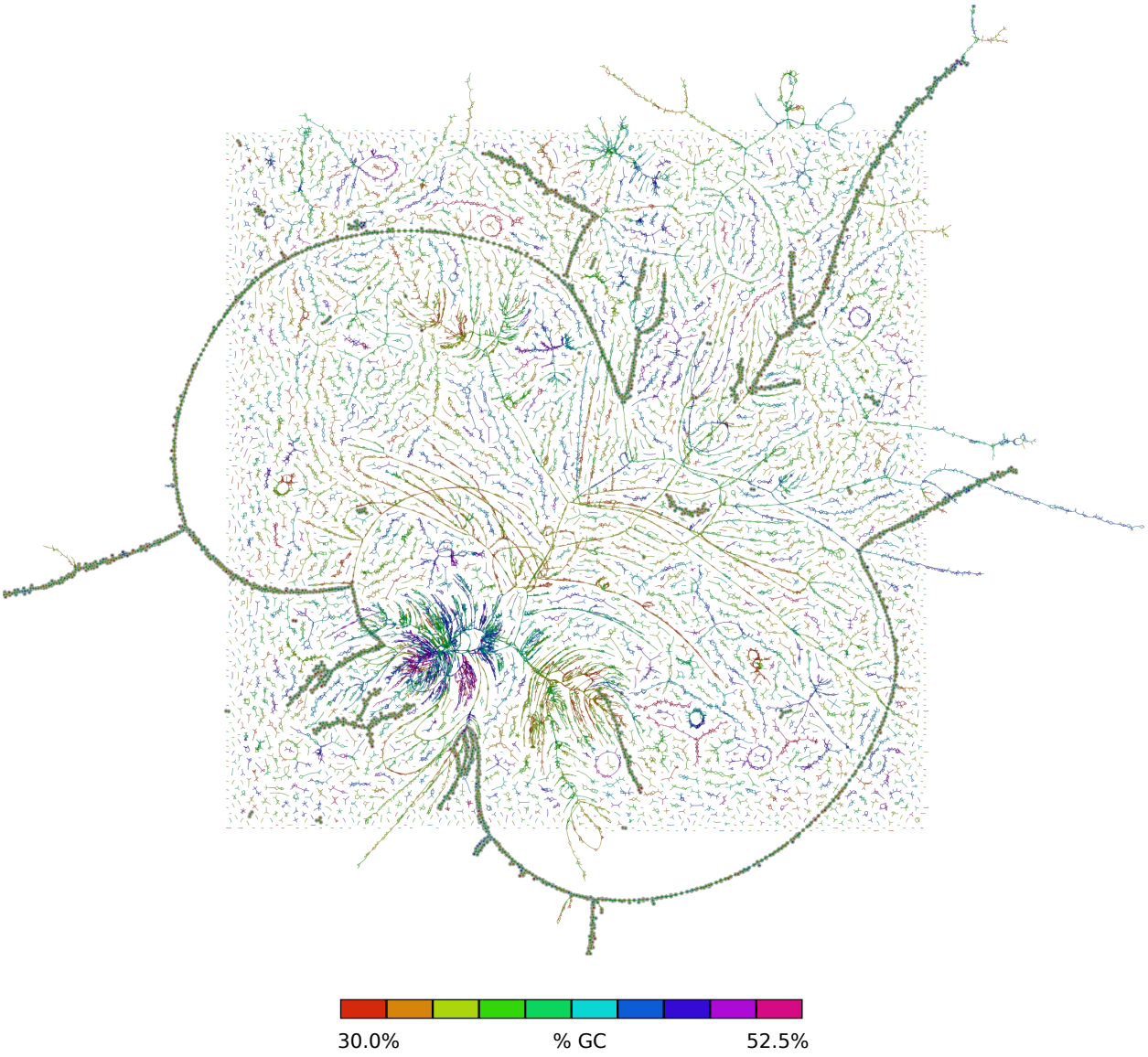

Figure A.9: Mate-pair connection graph illustrating the October 2008 metagenome *de novo* assembly. Lines represent contigs with mate-pair connections scoring greater than 400 bits (~75% of the assembly). Long strands represent prokaryote genome sequences and small circular strands show likely virus/plasmid sequences. Contigs in the candidate genome assembly related to alpha-*Proteobacterium* str. HTCC2255 are indicated (shaded in gray).

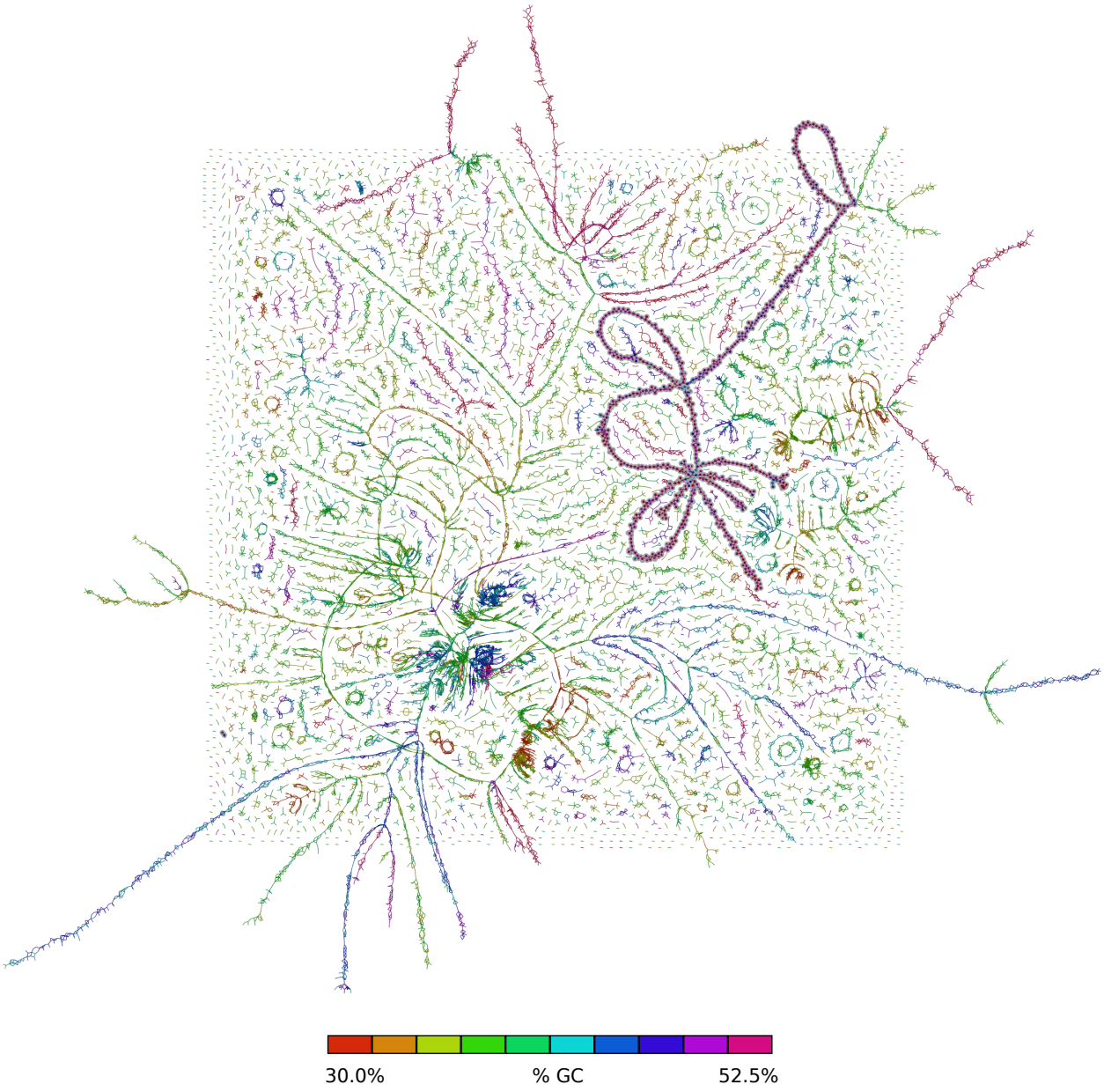

Figure A.10: Mate-pair connection graph illustrating the May 2009 metagenome *de novo* assembly. Lines represent contigs with mate-pair connections scoring greater than 750 bits (~60% of the assembly). Long strands represent prokaryote genome sequences and small circular strands show likely virus/plasmid sequences. Contigs aligning with the MG-II genome assembly are indicated (gray shading). This figure is a full-resolution representation of Figure 2.2.

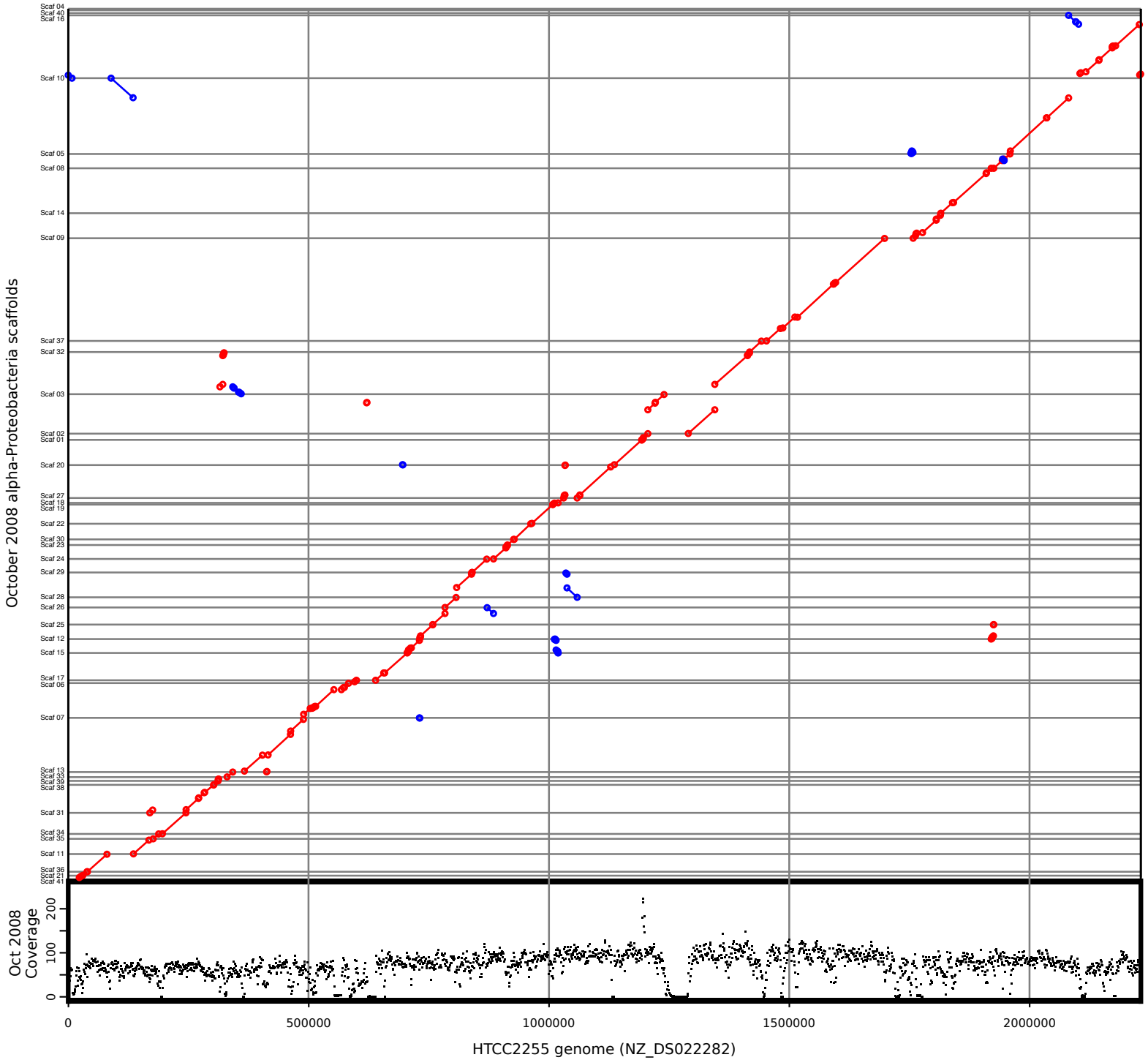
October 2008 alpha-Proteobacteria scaffolds
Scaf 04
Scaf 40
Scaf 16
Scaf 10
Scaf 05
Scaf 08
Scaf 14
Scaf 09
Scaf 37
Scaf 32
Scaf 03
Scaf 02
Scaf 01
Scaf 20
Scaf 27
Scaf 18
Scaf 19
Scaf 22
Scaf 30
Scaf 23
Scaf 24
Scaf 29
Scaf 28
Scaf 26
Scaf 25
Scaf 12
Scaf 15
Scaf 17
Scaf 06
Scaf 07
Scaf 13
Scaf 33
Scaf 39
Scaf 38
Scaf 31
Scaf 34
Scaf 35
Scaf 11
Scaf 36
Scaf 21
Scaf 41
Oct 2008 Coverage
0
100
200
0
500000
1000000
1500000
2000000
HTCC2255 genome (NZ_DS022282)

Figure A.11: Comparison of an October 2008 statistical cluster of assembled scaffolds to the genome of alpha-*Proteobacterium* str. HTCC2255. The x-axis represents coordinates of the HTCC2255 reference genome (NZ_DS022282). The y-axis of the upper plot shows coordinates of the 41 scaffolds automatically clustered together using tetra-nucleotide statistics (Figure A.9, shaded in gray), sorted and reversed as necessary to produce the most parsimonious alignment. Grey horizontal lines show scaffold boundaries. Diagonal lines in the upper plot show the extent of NUCMER nucleotide alignments between scaffolds and the reference genome, with red lines showing direct alignments and blue lines showing alignments of scaffold regions that were reversed relative to the primary scaffold alignment. The y-axis of the lower plot shows mean coverage (averaged over 1000bp segments) of the HTCC2255 reference genome by aligned October 2008 metagenomic reads.

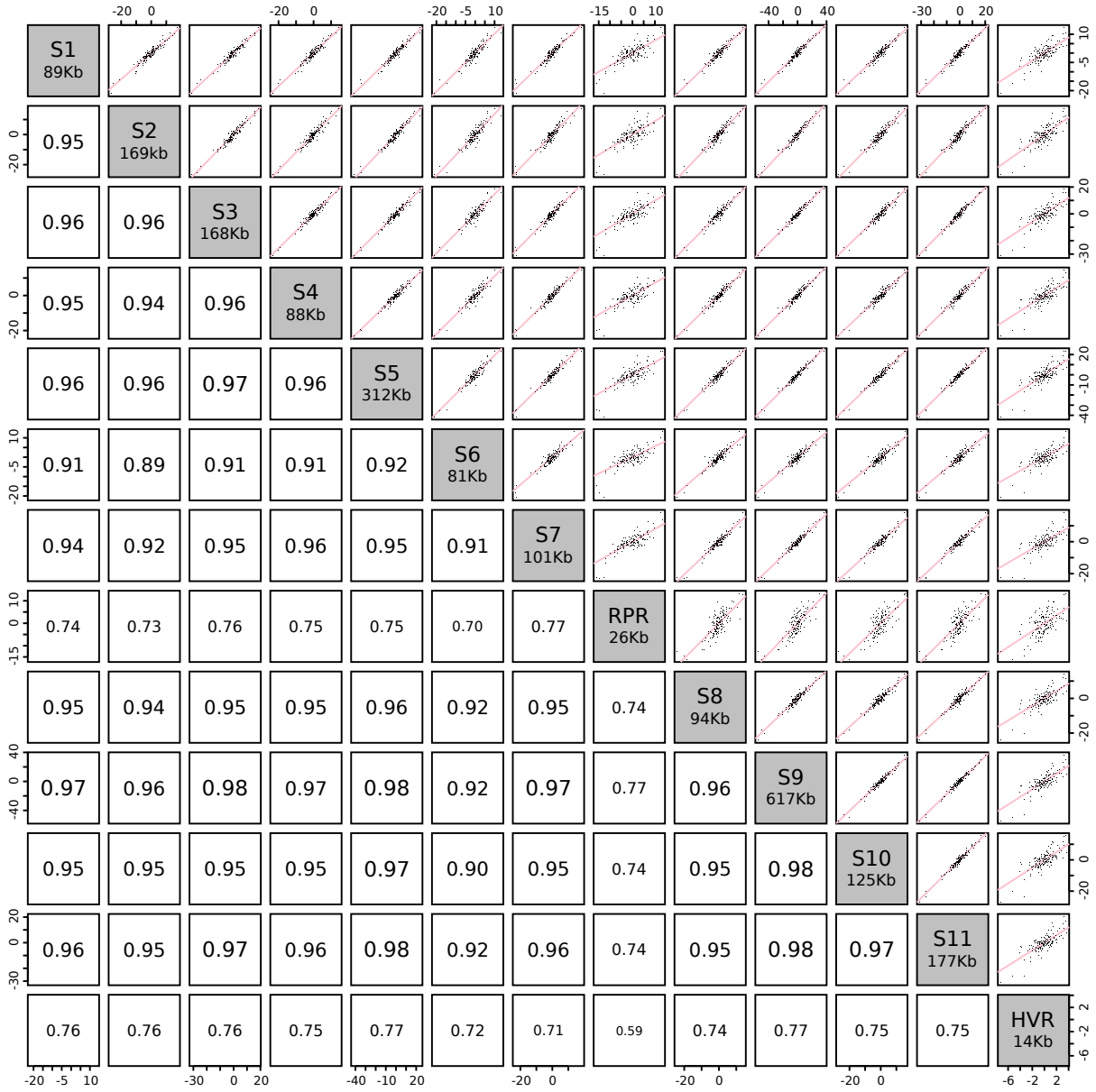

Figure A.12: Pairwise tetra-nucleotide usage anomaly correlations between scaffolds of the marine group II *Euryarchaeote* genome assembly. The eleven assembly scaffolds (S1 – S11) and the repeat region (RPR) and hyper-variable region (HVR) are shown on the matrix diagonal, with nucleotide sequence lengths noted. The upper half of the matrix shows pair-wise linear-regression plots of the tetra-nucleotide usage anomaly Z-statistics (x and y-axes, black dots, $n = 256$), with the pink-lines indicating the best linear model fits. The lower half of the matrix shows the correlation coefficients of the corresponding pair-wise regressions shown in the upper half.

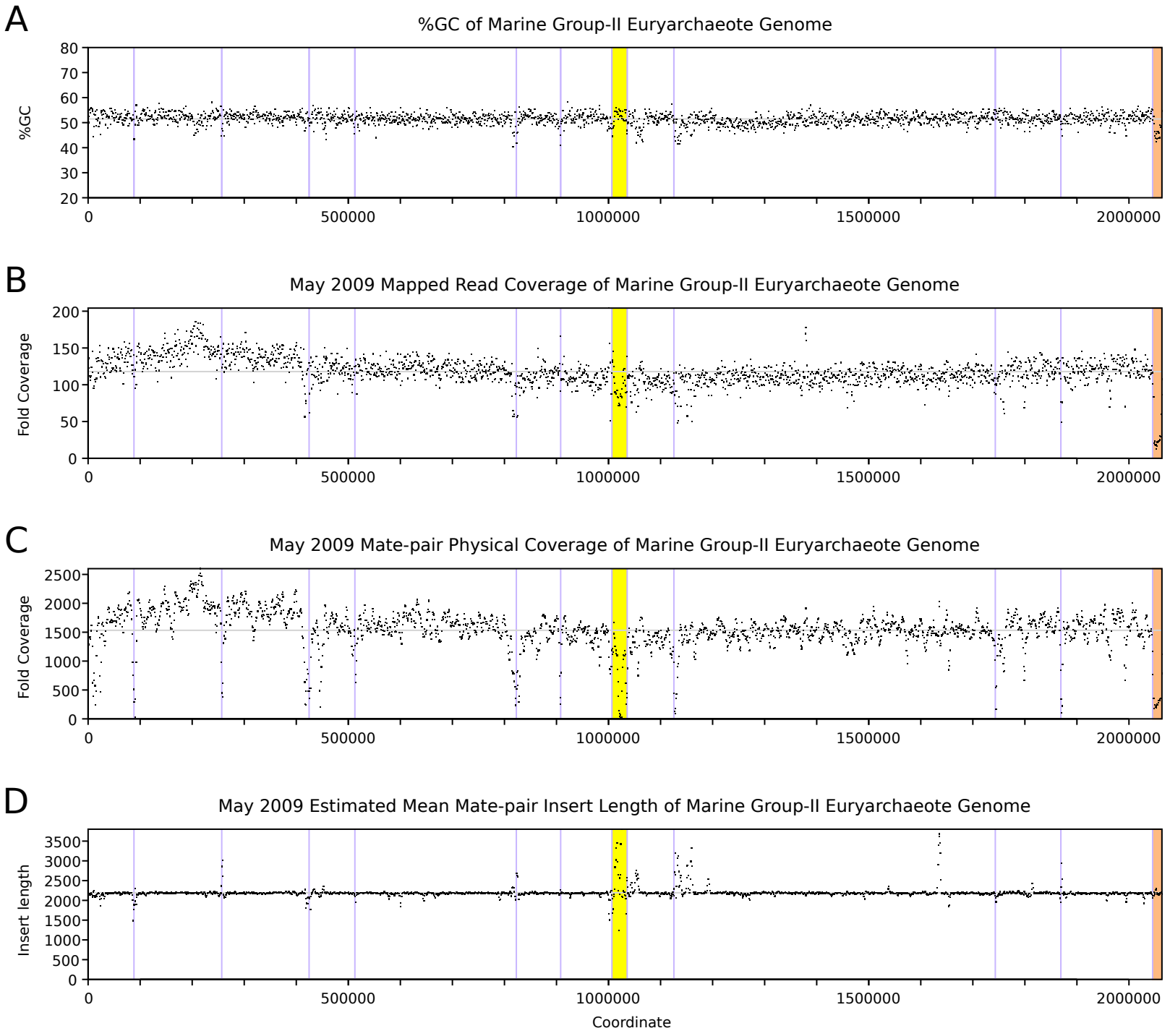
A
%GC of Marine Group-II Euryarchaeote Genome
%GC
80
70
60
50
40
30
20
0
500000
1000000
1500000
2000000
B
May 2009 Mapped Read Coverage of Marine Group-II Euryarchaeote Genome
Fold Coverage
200
150
100
50
0
0
500000
1000000
1500000
2000000
C
May 2009 Mate-pair Physical Coverage of Marine Group-II Euryarchaeote Genome
Fold Coverage
2500
2000
1500
1000
500
0
0
500000
1000000
1500000
2000000
D
May 2009 Estimated Mean Mate-pair Insert Length of Marine Group-II Euryarchaeote Genome
Insert length
3500
3000
2500
2000
1500
1000
500
0
0
500000
1000000
1500000
2000000
Coordinate

Figure A.13: Sequence statistics for the marine group II *Euryarchaeote* genome (MG-II). For all plots the x-axis corresponds to coordinates of MG-II, and values shown are averaged over 1000bp segments. Purple vertical lines denote scaffold boundaries, and the yellow and orange regions show the repeat region (RPR) and hyper-variable region (HVR), respectively. Horizontal grey lines show the genome-wide mean value for the plotted statistic.

(A) Mean %GC content of MG-II.

(B) Mean coverage of MG-II by aligned May 2009 metagenomic reads.

(C) Mean physical coverage of MG-II by aligned mate- paired reads from the May 2009 metagenome.

(D) Estimated mean mate-pair insert length for mate-pair alignments spanning positions along MG-II.

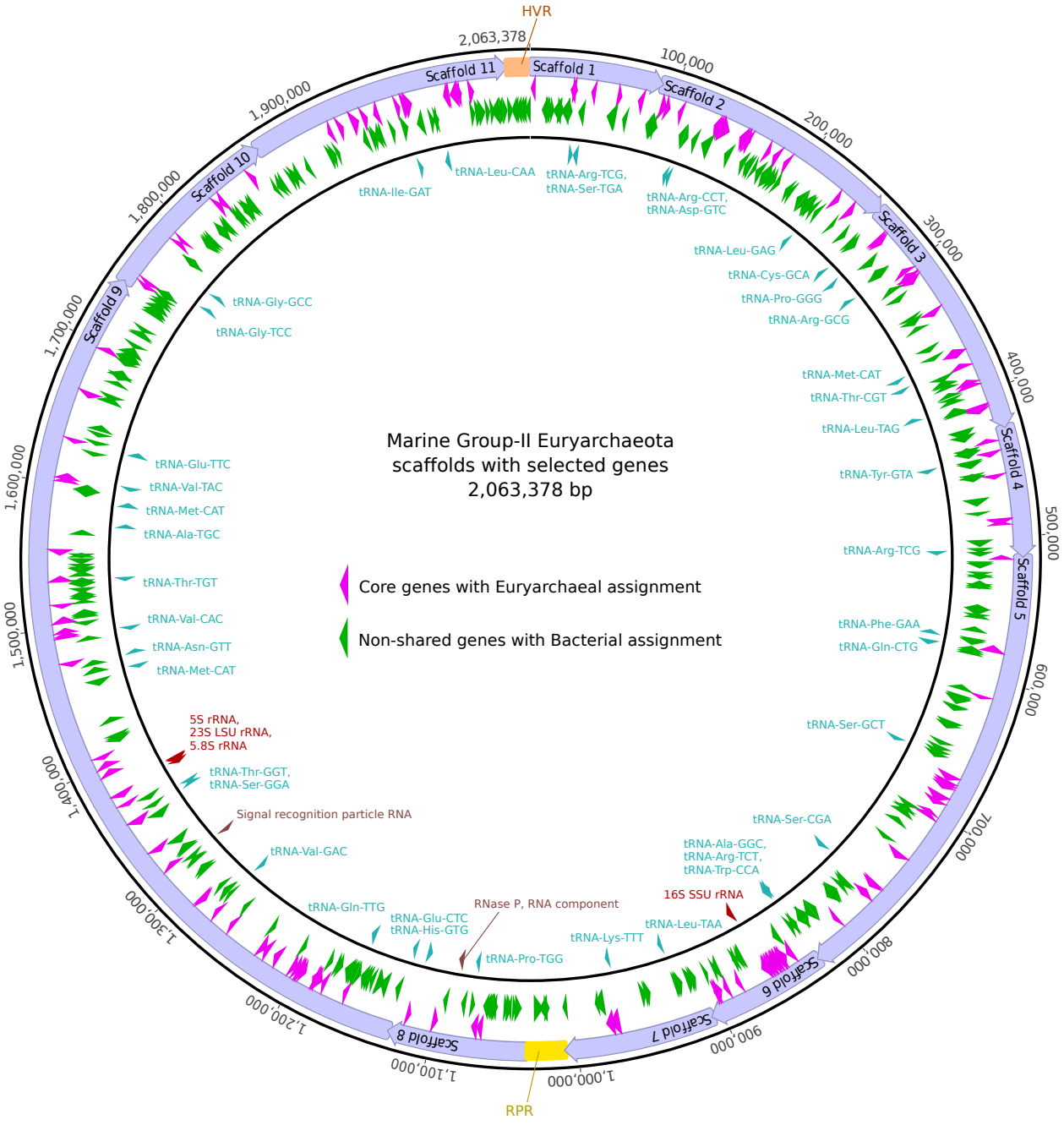
HVR
2,063,378
100,000
200,000
300,000
400,000
500,000
600,000
700,000
800,000
900,000
1,000,000
1,100,000
1,200,000
1,300,000
1,400,000
1,500,000
1,600,000
1,700,000
1,800,000
1,900,000
Scaffold 1
Scaffold 2
Scaffold 3
Scaffold 4
Scaffold 5
Scaffold 6
Scaffold 7
Scaffold 8
Scaffold 9
Scaffold 10
Scaffold 11
RPR
Marine Group-II Euryarchaeota
scaffolds with selected genes
2,063,378 bp
Core genes with Euryarchaeal assignment
Non-shared genes with Bacterial assignment
tRNA-Ile-GAT
tRNA-Leu-CAA
tRNA-Arg-TCG,
tRNA-Ser-TGA
tRNA-Arg-CCT,
tRNA-Asp-GTC
tRNA-Leu-GAG
tRNA-Cys-GCA
tRNA-Pro-GGG
tRNA-Arg-GCG
tRNA-Met-CAT
tRNA-Thr-CGT
tRNA-Leu-TAG
tRNA-Tyr-GTA
tRNA-Arg-TCG
tRNA-Phe-GAA
tRNA-Gln-CTG
tRNA-Ser-GCT
tRNA-Ser-CGA
tRNA-Ala-GGC,
tRNA-Arg-TCT,
tRNA-Trp-CCA
16S SSU rRNA
tRNA-Leu-TAA
tRNA-Lys-TTT
tRNA-Pro-TGG
RNase P, RNA component
tRNA-Gln-TTG
tRNA-Glu-CTC
tRNA-His-GTG
tRNA-Val-GAC
Signal recognition particle RNA
tRNA-Thr-GGT,
tRNA-Ser-GGA
5S rRNA,
23S LSU rRNA,
5.8S rRNA
tRNA-Met-CAT
tRNA-Asn-GTT
tRNA-Val-CAC
tRNA-Thr-TGT
tRNA-Ala-TGC
tRNA-Met-CAT
tRNA-Val-TAC
tRNA-Glu-TTC
tRNA-Gly-TCC
tRNA-Gly-GCC

Figure A.14: Schematic representation of the assembled circular marine group II *Euryarchaeote* genome. From the outside inwards the rings show: Genome coordinates, scaffold regions, “Core” gene models, “Bacterial” gene models, and ncRNA genes. RPR (yellow) and HVR (orange) denote the repeat and hyper-variable regions respectively. “Core” gene models (magenta) code for proteins homologous with proteins in both *Aciduliprofundum boonei* (NC_013926) and *Nitrosopumilus maritimus* (NC_010085) – based on reciprocal best blast hits – that are classified taxonomically, by MEGAN, within the phylum *Euryarchaeota*. “Bacterial” gene models (green) code for proteins without homologs in either *A. boonei* or *N. maritimus* that classify taxonomically within the domain Bacteria. Non-coding RNA genes include tRNAs (blue), rRNA (red), other ncRNAs (brown).

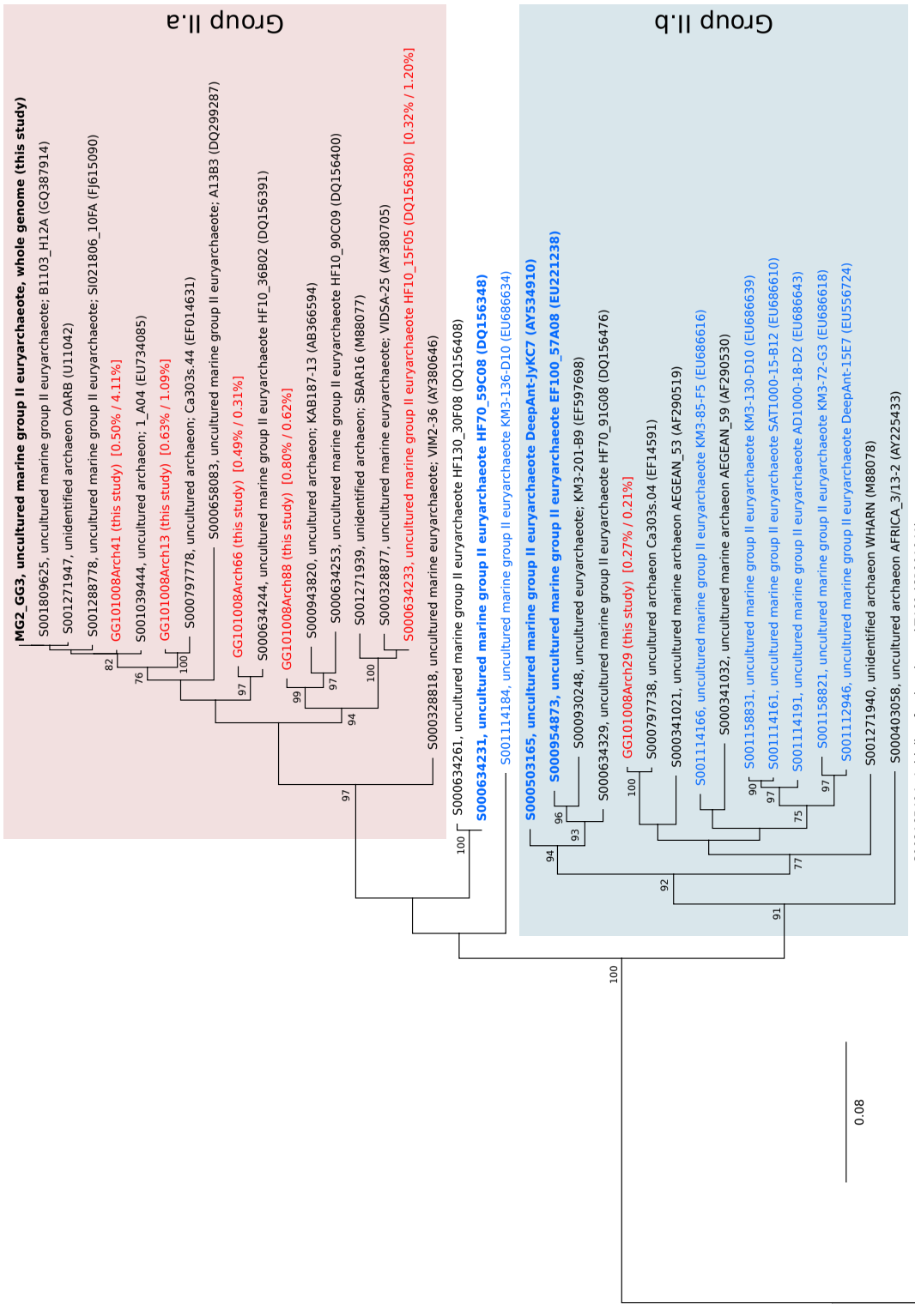
Group II.a
Group II.b
MG2_GG3, uncultured marine group II euryarchaeote, whole genome (this study)
S001809625, uncultured marine group II euryarchaeote; B1103_H12A (GQ387914)
S001271947, unidentified archaeon OARB (U11042)
S001288778, uncultured marine group II euryarchaeote; SI021806_10FA (FJ615090)
GG101008Arch41 (this study) [0.50% / 4.11%]
S001039444, uncultured archaeon; 1_A04 (EU734085)
GG101008Arch13 (this study) [0.63% / 1.09%]
S000797778, uncultured archaeon; Ca303s.44 (EF014631)
S000658083, uncultured marine group II euryarchaeote; A13B3 (DQ299287)
GG101008Arch66 (this study) [0.49% / 0.31%]
S000634244, uncultured marine group II euryarchaeote HF10_36B02 (DQ156391)
GG101008Arch88 (this study) [0.80% / 0.62%]
S000943820, uncultured archaeon; KAB187-13 (AB366594)
S000634253, uncultured marine group II euryarchaeote HF10_90C09 (DQ156400)
S001271939, unidentified archaeon; SBAR16 (M88077)
S000328877, uncultured marine euryarchaeote; VIDSA-25 (AY380705)
S000634233, uncultured marine group II euryarchaeote HF10_15F05 (DQ156380) [0.32% / 1.20%]
S000328818, uncultured marine euryarchaeote; VIM2-36 (AY380646)
S000634261, uncultured marine group II euryarchaeote HF130_30F08 (DQ156408)
S000634231, uncultured marine group II euryarchaeote HF70_59C08 (DQ156348)
S001114184, uncultured marine group II euryarchaeote KM3-136-D10 (EU686634)
S000503165, uncultured marine group II euryarchaeote DeepAnt-JyKC7 (AY534910)
S000954873, uncultured marine group II euryarchaeote EF100_57A08 (EU221238)
S000930248, uncultured euryarchaeote; KM3-201-B9 (EF597698)
S000634329, uncultured marine group II euryarchaeote HF70_91G08 (DQ156476)
GG101008Arch29 (this study) [0.27% / 0.21%]
S000797738, uncultured archaeon Ca303s.04 (EF14591)
S000341021, uncultured marine archaeon AEGEAN_53 (AF290519)
S001114166, uncultured marine group II euryarchaeote KM3-85-F5 (EU686616)
S000341032, uncultured marine archaeon AEGEAN_59 (AF290530)
S001158831, uncultured marine group II euryarchaeote KM3-130-D10 (EU686639)
S001114161, uncultured marine group II euryarchaeote SAT1000-15-B12 (EU686610)
S001114191, uncultured marine group II euryarchaeote AD1000-18-D2 (EU686643)
S001158821, uncultured marine group II euryarchaeote KM3-72-G3 (EU686618)
S001112946, uncultured marine group II euryarchaeote DeepAnt-15E7 (EU556724)
S001271940, unidentified archaeon WHARN (M88078)
S000403058, uncultured archaeon AFRICA_3/13-2 (AY225433)
S002165404, Aciduliprofundum boonei T469 (CP001941)
S001158827, uncultured marine group III euryarchaeote SAT1000-53-B3 (EU686632)
S000634332, uncultured marine group III euryarchaeote HF10_21C05 (DQ156479)
0.08

Figure A.15: Maximum likelihood phylogeny of full-length marine group II *Euryarchaeote* 16S rRNA sequences. The tree is rooted with *Aciduliprofundum boonei*. Branch lengths are proportional to the number of substitutions per site. Bootstraps are shown for 100 replicate trees. All taxa are identified by RDP ID, except those prefaced with "GG", which are October 2008 clones from this study, and the sequence for the marine group II *Euryarchaeote genome* (MG2_GG3) which is shown in black bold type. Taxa in blue type are from fully sequenced large-insert environmental clones, and those in bold blue type are shown in Figure 2.3B. Taxa in red type were selected from the October 16S clones + RDP database by the method used to estimate abundance from aligned metagenomic reads, and the estimated relative abundances attributed to each taxa for the October and May samples, respectively, are shown in square brackets. The red and blue regions show taxa that group within the two previously identified major 16S environmental clades within the marine group II *Euryarchaeota* (Groups II.a and II.b).

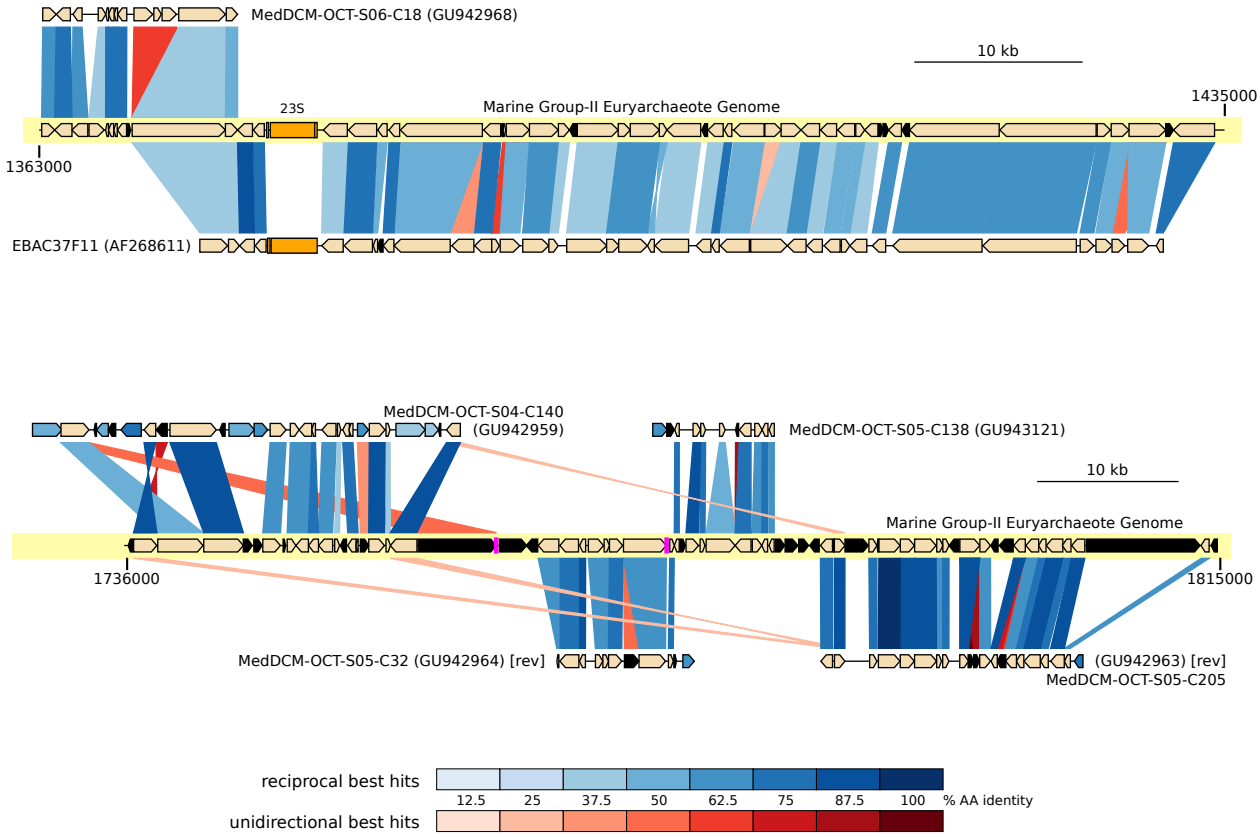

Figure A.16: Comparison of homologous proteins between environmental clones and two additional regions of the marine group II *Euryarchaeote* genome (MG-II). MG-II regions correspond to grey shaded regions shown on Figure 2.3A. Homologous genes are connected by shaded regions; blue and red indicate reciprocal and unidirectional best hits, and the depth of shading indicates percent amino acid identity on the scale shown. Gene models, depicted as pointed boxes, are shaded black for genes with no blast hits, blue for environmental genes with a reciprocal best hit to an undepicted location in MG-II, and beige otherwise. rRNA and tRNA genes are shown in orange and magenta, respectively. Clone identifiers and accession numbers are shown for environmental sequences, and those marked "[rev]" are depicted reversed relative to the deposited sequence.

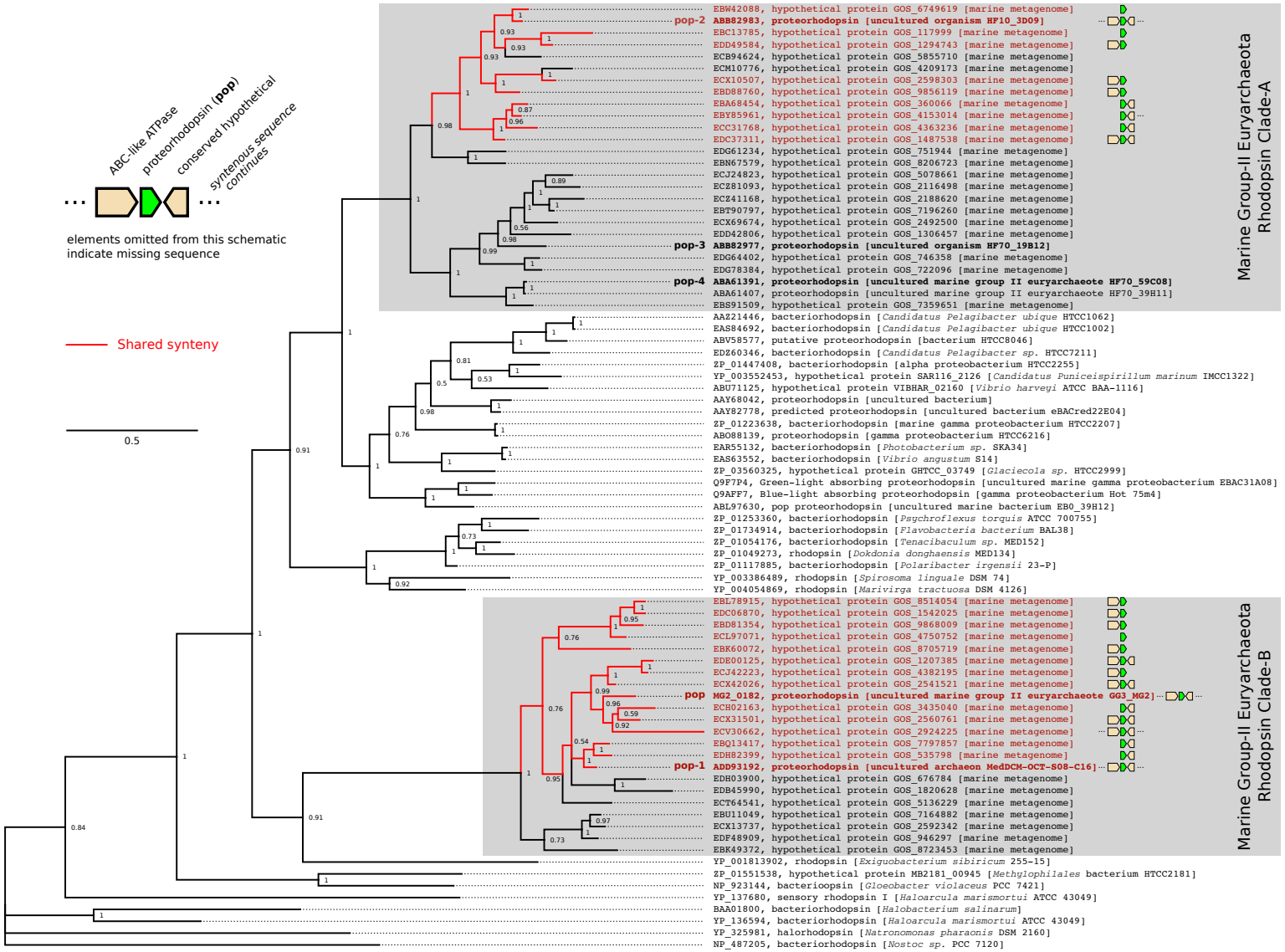
ABC-like ATPase
proteorhodopsin (pop)
conserved hypothetical
syntenous sequence
continues
elements omitted from this schematic
indicate missing sequence
Shared synteny
0.5
pop-2
pop-3
pop-4
pop
pop-1
Marine Group-II Euryarchaeota
Rhodopsin Clade-A
Marine Group-II Euryarchaeota
Rhodopsin Clade-B
EBW42088, hypothetical protein GOS_6749619 [marine metagenome]
ABB82983, proteorhodopsin [uncultured organism HF10_3D09]
EBC13785, hypothetical protein GOS_117999 [marine metagenome]
EDD49584, hypothetical protein GOS_1294743 [marine metagenome]
ECB94624, hypothetical protein GOS_5855710 [marine metagenome]
ECM10776, hypothetical protein GOS_4209173 [marine metagenome]
ECX10507, hypothetical protein GOS_2598303 [marine metagenome]
EBD88760, hypothetical protein GOS_9856119 [marine metagenome]
EBA68454, hypothetical protein GOS_360066 [marine metagenome]
EBY85961, hypothetical protein GOS_4153014 [marine metagenome]
ECC31768, hypothetical protein GOS_4363236 [marine metagenome]
EDC37311, hypothetical protein GOS_1487538 [marine metagenome]
EDG61234, hypothetical protein GOS_751944 [marine metagenome]
EBN67579, hypothetical protein GOS_8206723 [marine metagenome]
ECJ24823, hypothetical protein GOS_5078661 [marine metagenome]
EC281093, hypothetical protein GOS_2116498 [marine metagenome]
ECZ41168, hypothetical protein GOS_2188620 [marine metagenome]
EBT90797, hypothetical protein GOS_7196260 [marine metagenome]
ECX69674, hypothetical protein GOS_2492500 [marine metagenome]
EDD42806, hypothetical protein GOS_1306457 [marine metagenome]
ABB82977, proteorhodopsin [uncultured organism HF70_19B12]
EDG64402, hypothetical protein GOS_746358 [marine metagenome]
EDG78384, hypothetical protein GOS_722096 [marine metagenome]
ABA61391, proteorhodopsin [uncultured marine group II euryarchaeote HF70_59C08]
ABA61407, proteorhodopsin [uncultured marine group II euryarchaeote HF70_39H11]
EBS91509, hypothetical protein GOS_7359651 [marine metagenome]
AAZ21446, bacteriorhodopsin [Candidatus Pelagibacter ubique HTCC1062]
EAS84692, bacteriorhodopsin [Candidatus Pelagibacter ubique HTCC1002]
ABV58577, putative proteorhodopsin [bacterium HTCC8046]
EDZ60346, bacteriorhodopsin [Candidatus Pelagibacter sp. HTCC7211]
ZP_01447408, bacteriorhodopsin [alpha proteobacterium HTCC2255]
YP_003552453, hypothetical protein SAR116_2126 [Candidatus Puniceispirillum marinum IMCC1322]
ABU71125, hypothetical protein VIBHAR_02160 [Vibrio harveyi ATCC BAA-1116]
AAY68042, proteorhodopsin [uncultured bacterium]
AAY82778, predicted proteorhodopsin [uncultured bacterium eBACred22E04]
ZP_01223638, bacteriorhodopsin [marine gamma proteobacterium HTCC2207]
ABO88139, proteorhodopsin [gamma proteobacterium HTCC6216]
EAR55132, bacteriorhodopsin [Photobacterium sp. SKA34]
EAS63552, bacteriorhodopsin [Vibrio angustum S14]
ZP_03560325, hypothetical protein GHTCC_03749 [Glaciecola sp. HTCC2999]
Q9F7P4, Green-light absorbing proteorhodopsin [uncultured marine gamma proteobacterium EBAC31A08]
Q9AFF7, Blue-light absorbing proteorhodopsin [gamma proteobacterium Hot 75m4]
ABL97630, pop proteorhodopsin [uncultured marine bacterium EB0_39H12]
ZP_01253360, bacteriorhodopsin [Psychroflexus torquis ATCC 700755]
ZP_01734914, bacteriorhodopsin [Flavobacteria bacterium BAL38]
ZP_01054176, bacteriorhodopsin [Tenacibaculum sp. MED152]
ZP_01049273, rhodopsin [Dokdonia donghaensis MED134]
ZP_01117885, bacteriorhodopsin [Polaribacter irgensii 23-P]
YP_003386489, rhodopsin [Spirosoma linguale DSM 74]
YP_004054869, rhodopsin [Marivirga tractuosa DSM 4126]
EBL78915, hypothetical protein GOS_8514054 [marine metagenome]
EDC06870, hypothetical protein GOS_1542025 [marine metagenome]
EBD81354, hypothetical protein GOS_9868009 [marine metagenome]
ECL97071, hypothetical protein GOS_4750752 [marine metagenome]
EBK60072, hypothetical protein GOS_8705719 [marine metagenome]
EDE00125, hypothetical protein GOS_1207385 [marine metagenome]
ECJ42223, hypothetical protein GOS_4382195 [marine metagenome]
ECX42026, hypothetical protein GOS_2541521 [marine metagenome]
MG2_0182, proteorhodopsin [uncultured marine group II euryarchaeote GG3_MG2]
ECH02163, hypothetical protein GOS_3435040 [marine metagenome]
ECX31501, hypothetical protein GOS_2560761 [marine metagenome]
ECV30662, hypothetical protein GOS_2924225 [marine metagenome]
EBQ13417, hypothetical protein GOS_7797857 [marine metagenome]
EDH82399, hypothetical protein GOS_535798 [marine metagenome]
ADD93192, proteorhodopsin [uncultured archaeon MedDCM-OCT-S08-C16]
EDH03900, hypothetical protein GOS_676784 [marine metagenome]
EDB45990, hypothetical protein GOS_1820628 [marine metagenome]
ECT64541, hypothetical protein GOS_5136229 [marine metagenome]
EBU11049, hypothetical protein GOS_7164882 [marine metagenome]
ECX13737, hypothetical protein GOS_2592342 [marine metagenome]
EDF48909, hypothetical protein GOS_946297 [marine metagenome]
EBK49372, hypothetical protein GOS_8723453 [marine metagenome]
YP_001813902, rhodopsin [Exiguobacterium sibiricum 255-15]
ZP_01551538, hypothetical protein MB2181_00945 [Methylophilales bacterium HTCC2181]
NP_923144, bacterioopsin [Gloeobacter violaceus PCC 7421]
YP_137680, sensory rhodopsin I [Haloarcula marismortui ATCC 43049]
BAA01800, bacteriorhodopsin [Halobacterium salinarum]
YP_136594, bacteriorhodopsin [Haloarcula marismortui ATCC 43049]
YP_325981, halorhodopsin [Natronomonas pharaonis DSM 2160]
NP_487205, bacteriorhodopsin [Nostoc sp. PCC 7120]

Figure A.17: Bayesian phylogeny of rhodopsin protein sequences. The tree is rooted with sequence YP_325981, the halorhodopsin from *Natronomonas pharaonis*. Branch lengths are proportional to the number of substitutions per site. Bayesian p-values are shown for each branch. All taxa are identified by NCBI accession number, except "MG2_0182", which is the locus id for the rhodopsin found in the marine group II *Euryarchaeote* genome (MG-II). Taxa in bold type and prefaced by a "pop" identifier correspond with those labeled in Figure 2.4. The two clades of marine group II *Euryarchaeote* rhodopsin proteins noted in Figure 2.4 are shown with grey shading. Red branches and leaves on the tree show marine group II *Euryarchaeote* rhodopsins of both types that appear in a syntenic context consistent with that seen in MG-II. The gene model schematic for each red taxa shows which shared neighboring gene models are present in the corresponding nucleotide sequence. Gene models omitted from a schematic indicate missing nucleotide sequence.

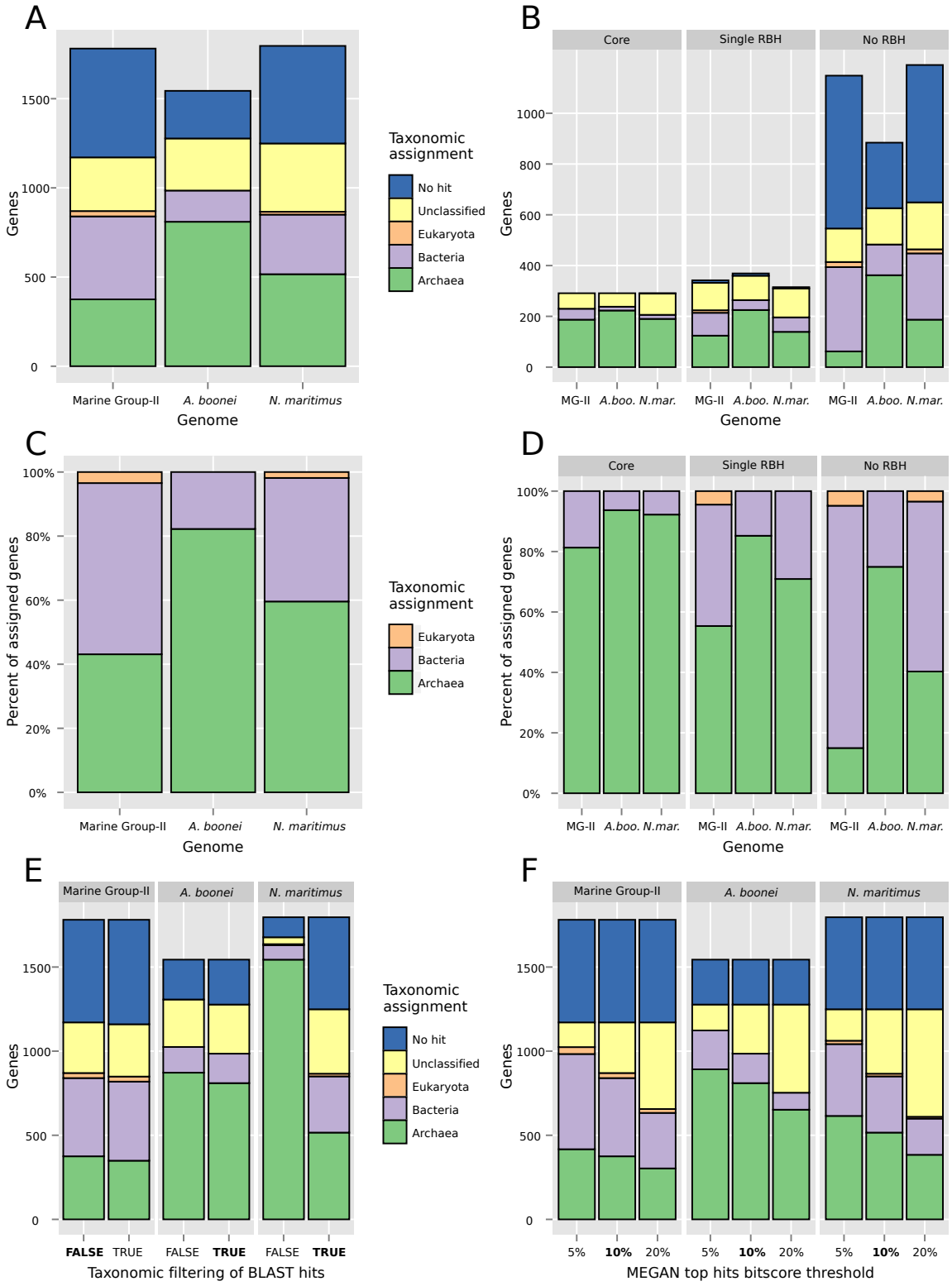
A
Genes
1500
1000
500
0
Marine Group-II
A. boonei
N. maritimus
Genome
Taxonomic assignment
No hit
Unclassified
Eukaryota
Bacteria
Archaea
B
Core
Single RBH
No RBH
Genes
1000
800
600
400
200
0
MG-II
A.boo.
N.mar.
Genome
C
Percent of assigned genes
100%
80%
60%
40%
20%
0%
Marine Group-II
A. boonei
N. maritimus
Genome
Taxonomic assignment
Eukaryota
Bacteria
Archaea
D
Core
Single RBH
No RBH
Percent of assigned genes
100%
80%
60%
40%
20%
0%
MG-II
A.boo.
N.mar.
Genome
E
Marine Group-II
A. boonei
N. maritimus
Genes
1500
1000
500
0
FALSE
TRUE
Taxonomic filtering of BLAST hits
Taxonomic assignment
No hit
Unclassified
Eukaryota
Bacteria
Archaea
F
Marine Group-II
A. boonei
N. maritimus
Genes
1500
1000
500
0
5%
10%
20%
MEGAN top hits bitscore threshold

Figure A.18: Details of the taxonomic assignment of Archaeal proteins, based on analysis of blast hits to the RefSeq protein database using MEGAN.

(A) Comparison of the taxonomic assignments of proteins from the marine group II *Euryarchaeote* genome, *Aciduliprofundum boonei* (NC_013926) and *Nitrosopumilus maritimus* (NC_010085). The y-axis shows a count of genes. Colored bars correspond to taxonomic classification of proteins by MEGAN. "No hit" indicates proteins with no blast homologs in the RefSeq database, and "Unclassified" indicates proteins that MEGAN could not place within a specific taxonomic domain.

(B) Comparison of the taxonomic assignments of proteins from the three reference genomes, further categorized into one of three groups via reciprocal best hit (RBH) blast analysis as shown in Figure 2.5B. Core proteins are those with symmetrical RBHs among all three genomes. Single RBH are those proteins that have RBH homologs within only two of the genomes, and No RBH are those without shared homologs.

(C) Relative comparison of the taxonomically assigned proteins from the three comparison genomes. The y-axis shows the percentage of all proteins coded by a genome that MEGAN could place within a specific taxonomic domain, as indicated by the colored bars.

(D) Relative comparison of the taxonomically assigned proteins from the three comparison genomes, further categorized into one of three groups as described for (B).

(E) Comparison of the effect of taxonomic filtering of the RefSeq database on taxonomic assignments. The x-axis categories "FALSE" and "TRUE", indicate whether blast hits to selected RefSeq taxa (other than the reference genome itself, which is always removed) were excluded from the MEGAN analysis. Bold type indicates the setting selected for use for all other MEGAN analyses in this study. See Methods for a list of RefSeq taxa excluded for each reference genome.

(F) Comparison of the effect of varying the "Top Percent" parameter of MEGAN's lowest common ancestor (LCA) algorithm. The x-axis categories 5%, 10% and 20% show the three parameter values evaluated. Bold type indicates the setting selected for use for all other MEGAN analyses in this study.

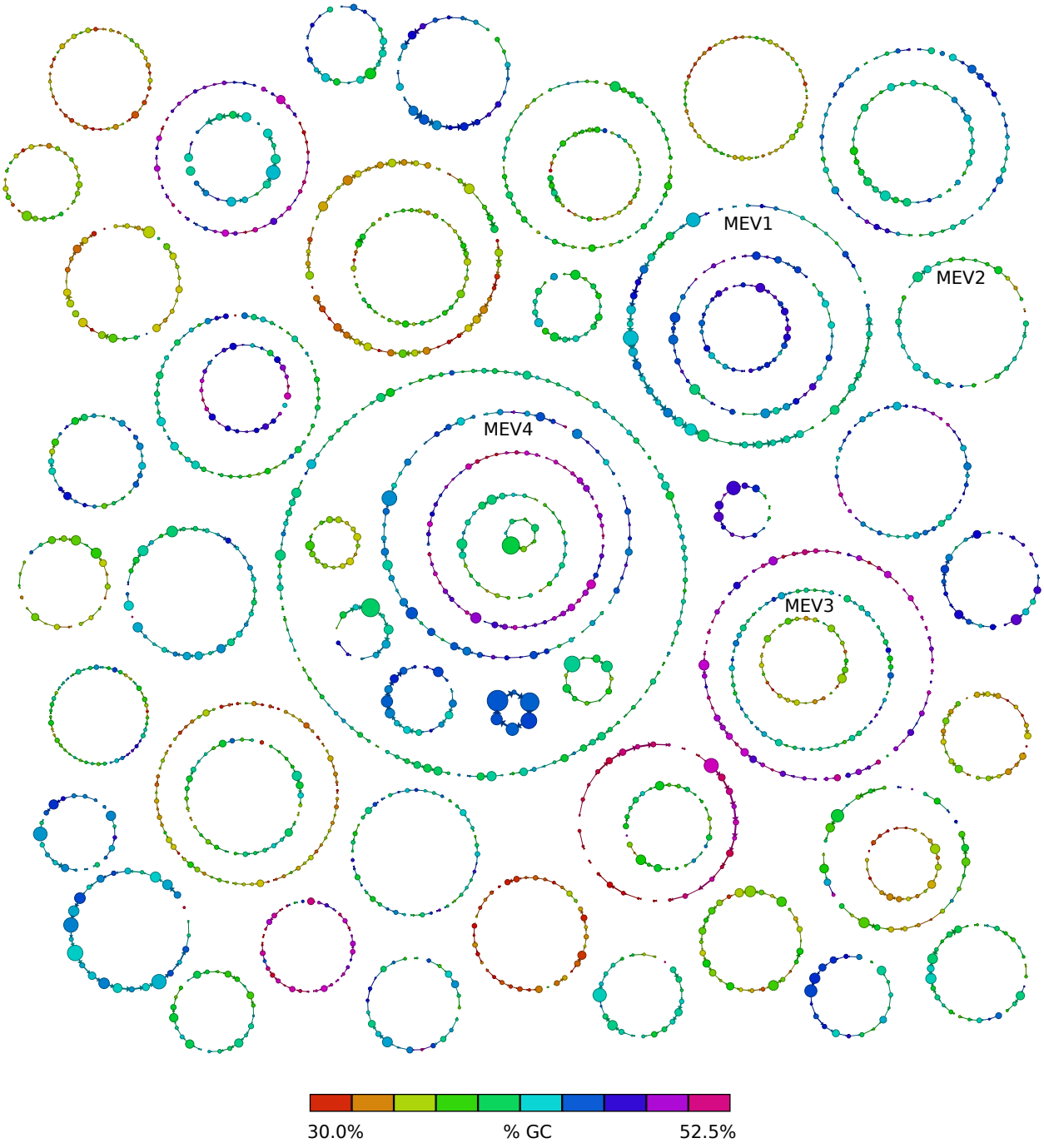

Figure A.19: Circular scaffolds assembled from the October 2008 sample. Mate-pair connection graph illustrating the resulting 59 scaffolds (see Table B.13). Small filled circles represent individual contigs, with GC content on the color scale shown, and area proportional to contig length. Marine Euryarchaeal viruses are labeled (MEV1–4).

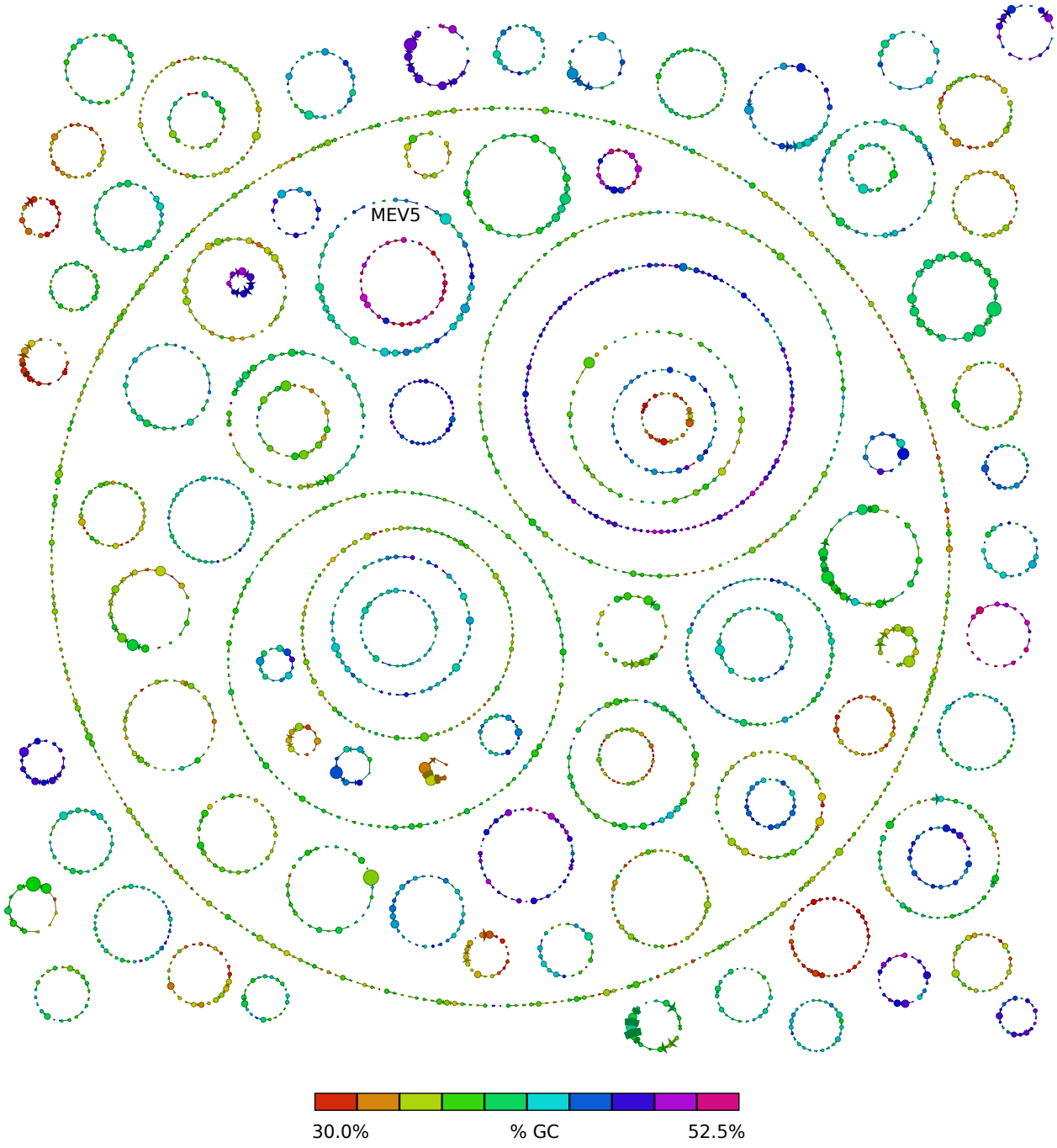

Figure A.20: Circular scaffolds assembled from the May 2009 sample. Mate-pair connection graph illustrating the resulting 91 scaffolds (see Table B.14). Small filled circles represent individual contigs, with GC content on the color scale shown, and area proportional to contig length. A marine Euryarchaeal virus is labeled (MEV5).

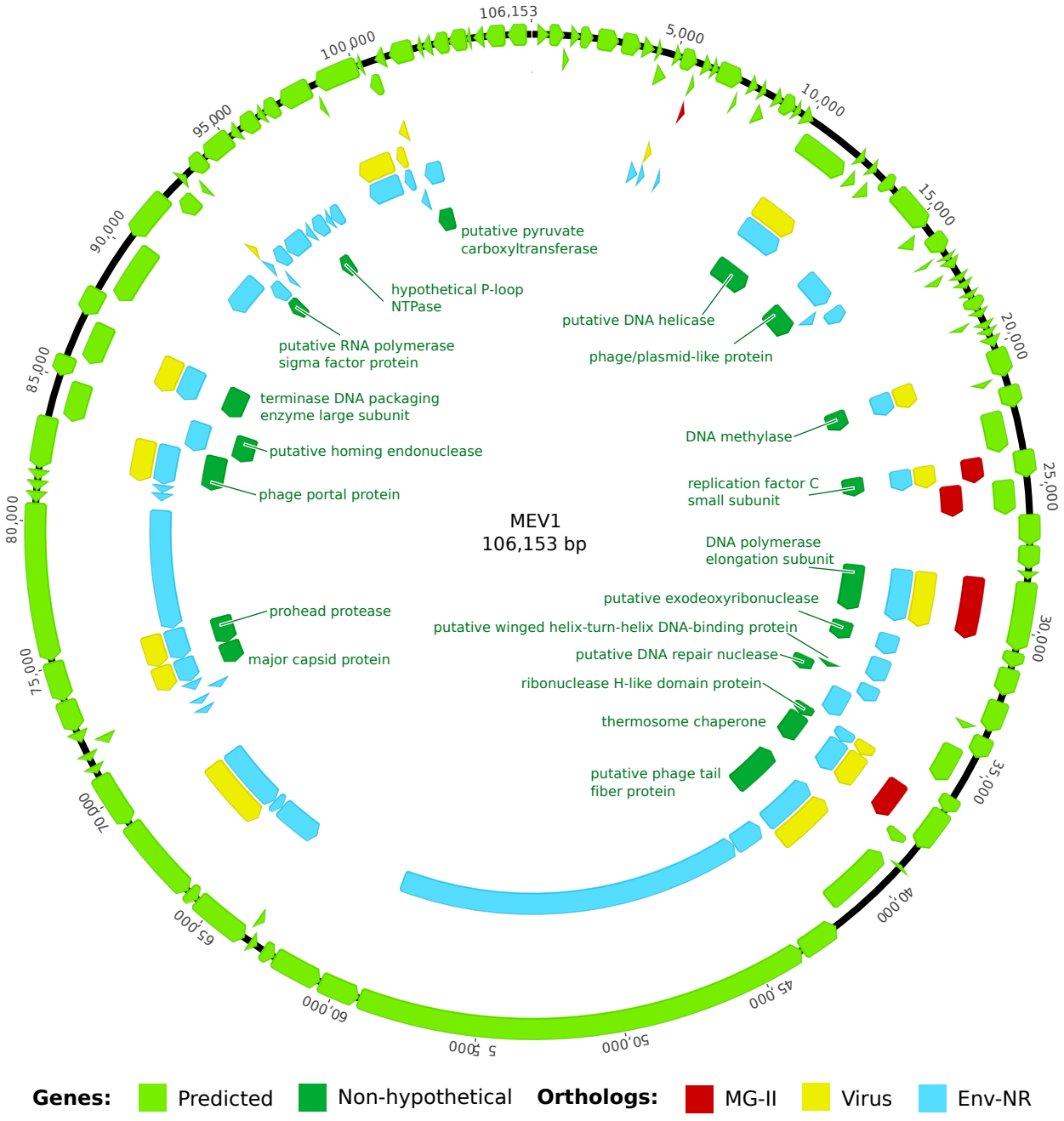

Figure A.21: Schematic representation of the assembled Group 1 virus genome MEV1. From the outside inwards the rings show: Genome coordinates, predicted genes (light green), genes with orthologs in the marine group II *Euryarchaeote* genome (CM001443, red), genes with orthologs in virus genomes in the NCBI RefSeq database (yellow), genes with orthologs in the NCBI "env-nr" protein database (blue, primarily from the Global Ocean Sampling project), and genes with non-hypothetical annotations assigned, as noted (dark green, see Table B.16). Orthologous genes were determined from BLAST hits to the indicated databases with e-values greater than 10^{-5}.

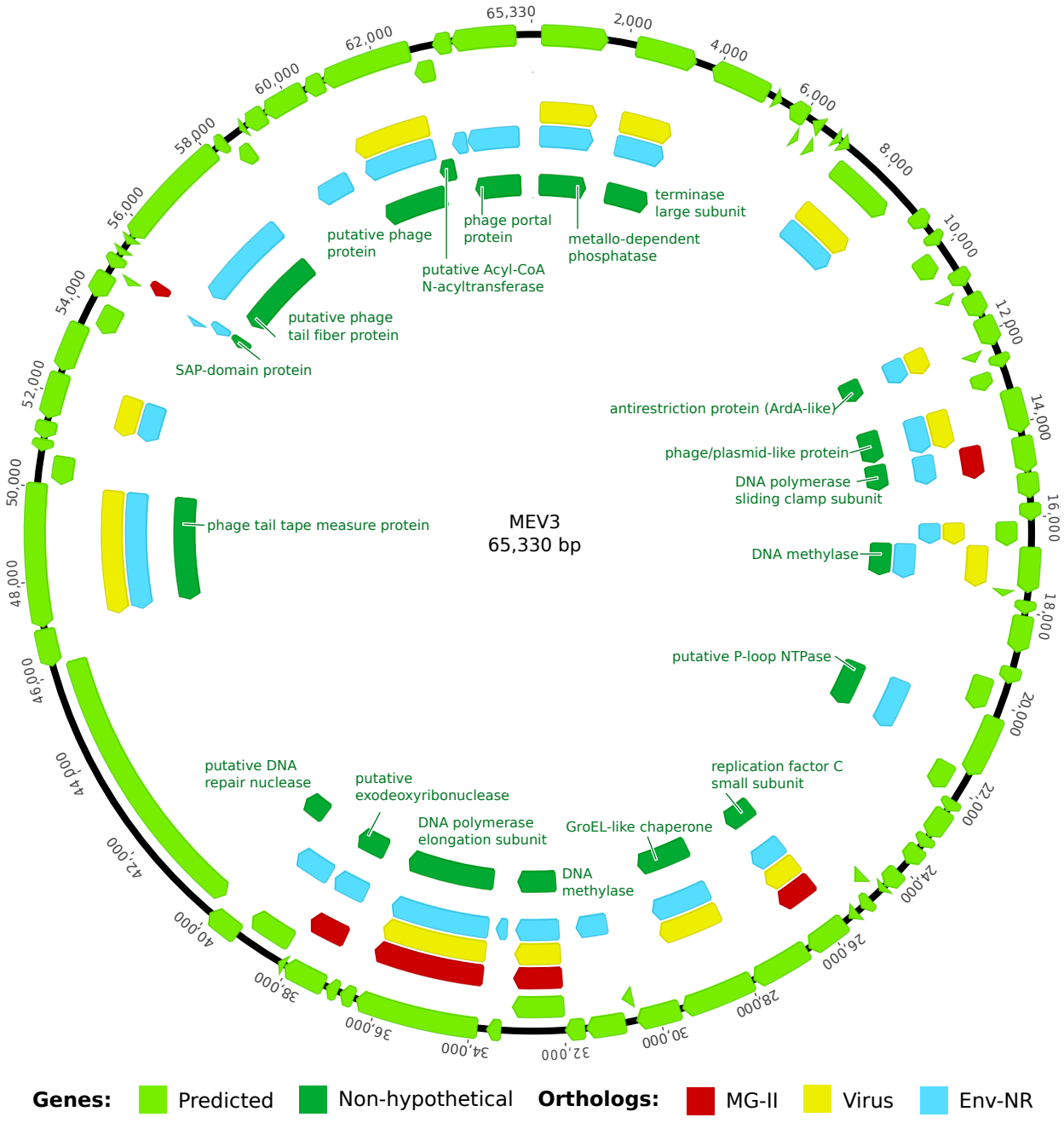

Figure A.22: Schematic representation of the assembled Group 2 virus genome MEV3. From the outside inwards the rings show: Genome coordinates, predicted genes (light green), genes with orthologs in the marine group II *Euryarchaeote* genome (CM001443, red), genes with orthologs in virus genomes in the NCBI RefSeq database (yellow), genes with orthologs in the NCBI "env-nr" protein database (blue, primarily from the Global Ocean Sampling project), and genes with non-hypothetical annotations assigned, as noted (dark green, see Table B.17). Orthologous genes were determined from BLAST hits to the indicated databases with e-values greater than 10^{-5}.

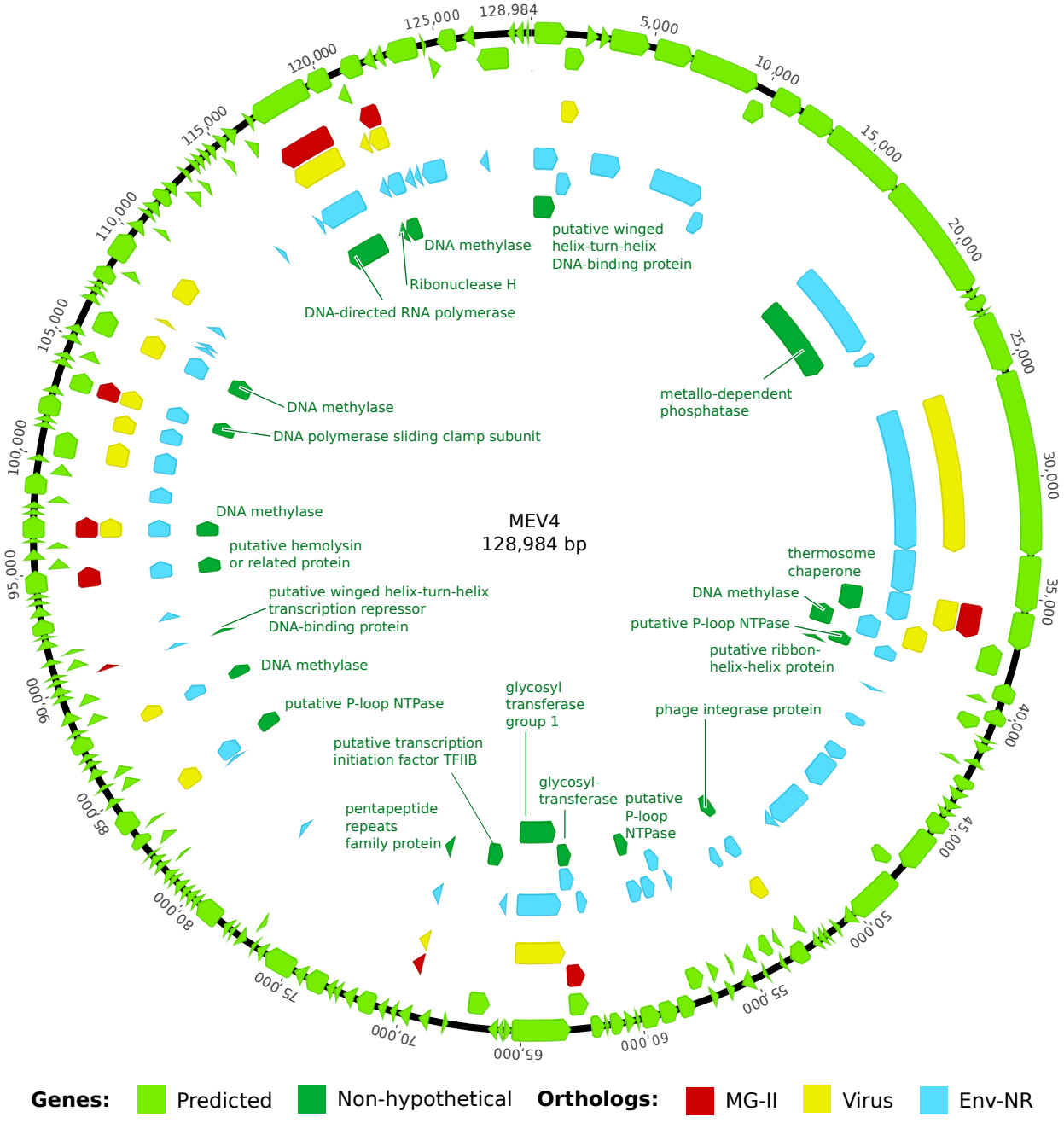

Figure A.23: Schematic representation of the assembled Group 3 virus genome MEV4. From the outside inwards the rings show: Genome coordinates, predicted genes (light green), genes with orthologs in the marine group II *Euryarchaeote* genome (CM001443, red), genes with orthologs in virus genomes in the NCBI RefSeq database (yellow), genes with orthologs in the NCBI "env-nr" protein database (blue, primarily from the Global Ocean Sampling project), and genes with non-hypothetical annotations assigned, as noted (dark green, see Table B.18). Orthologous genes were determined from BLAST hits to the indicated databases with e-values greater than 10^{-5}.

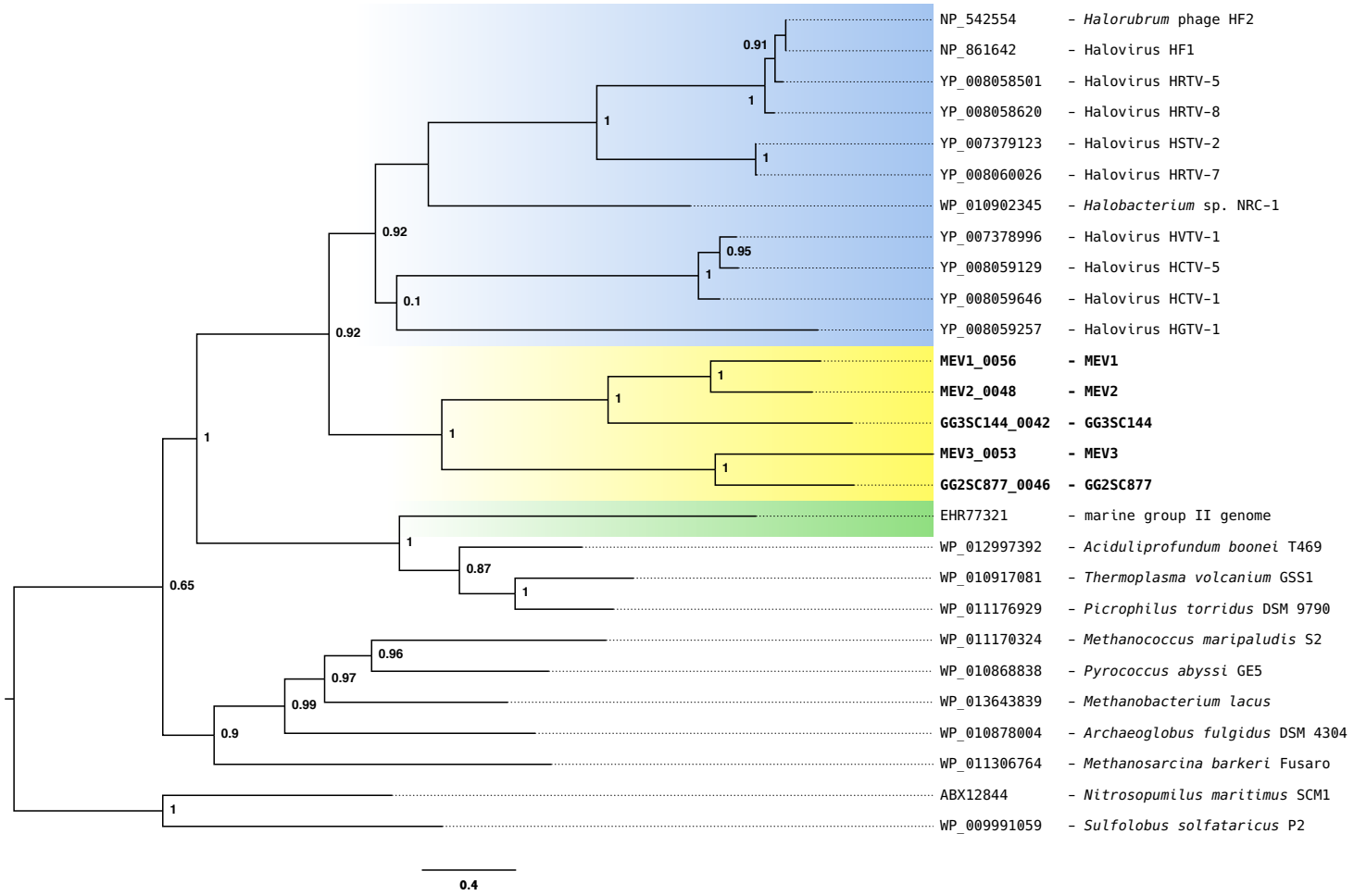

Figure A.24: Phylogeny of selected archaeal and virus DNA Polymerase elongation protein sequences. Highlighted are: Halovirus and selected Haloarchaeal sequences (blue), MEV Group 1 and 2 genome and scaffold sequences (yellow) and marine group II *Euryarchaeote* genomic sequence (green). NCBI protein accessions are indicated except for sequences from this study (bold type) which show locus identifiers. FastTree support values are shown for each branch. Branch lengths represent estimated substitutions per site with the scale shown.

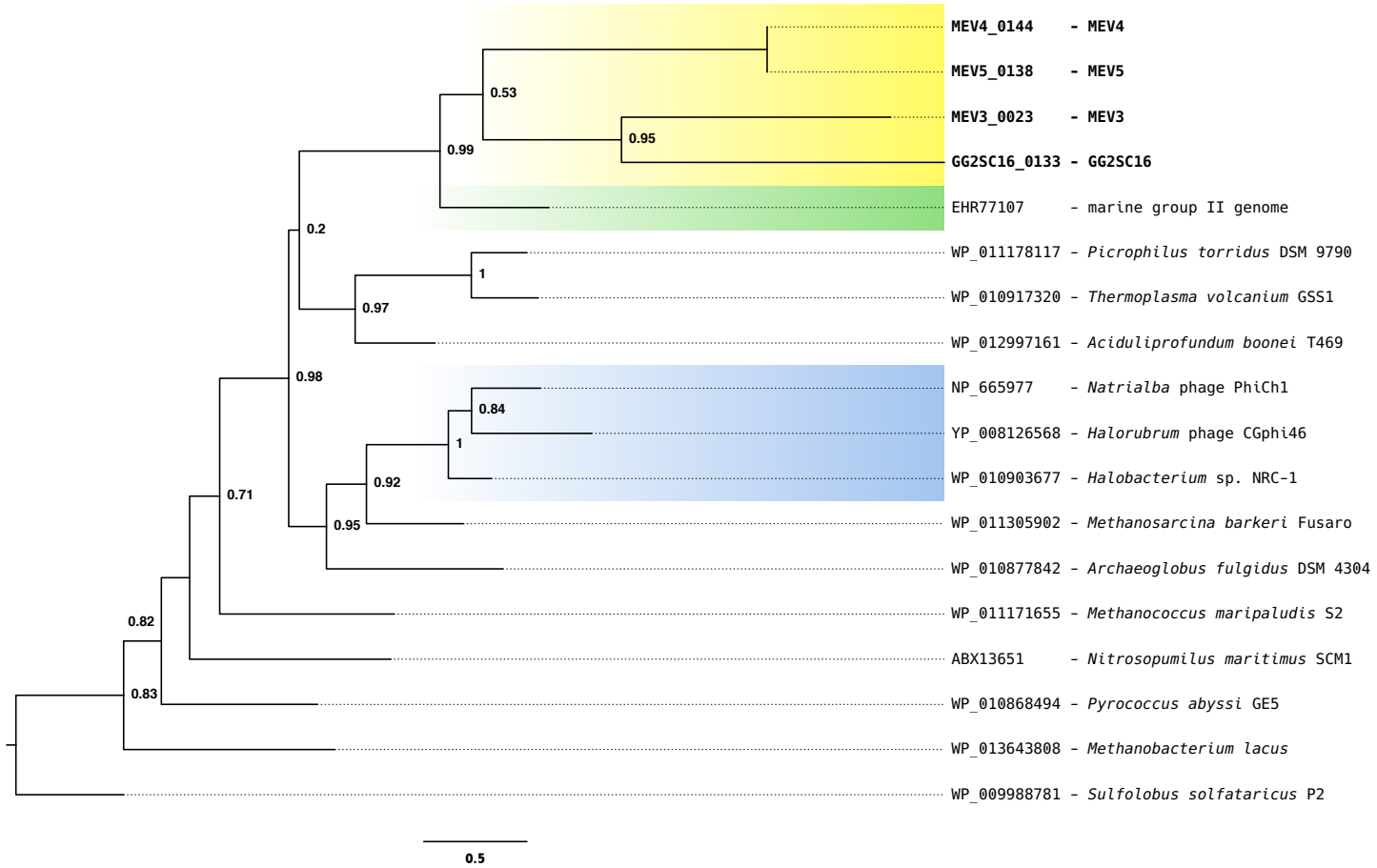

Figure A.25: Phylogeny of selected archaeal and virus DNA sliding clamp subunit protein sequences. Highlighted are: Halovirus and selected Haloarchaeal sequences (blue), MEV Group 2 and 3 genome and scaffold sequences (yellow) and marine group II *Euryarchaeote* genomic sequence (green). NCBI protein accessions are indicated except for sequences from this study (bold type) which show locus identifiers. FastTree support values are shown for each branch. Branch lengths represent estimated substitutions per site with the scale shown.

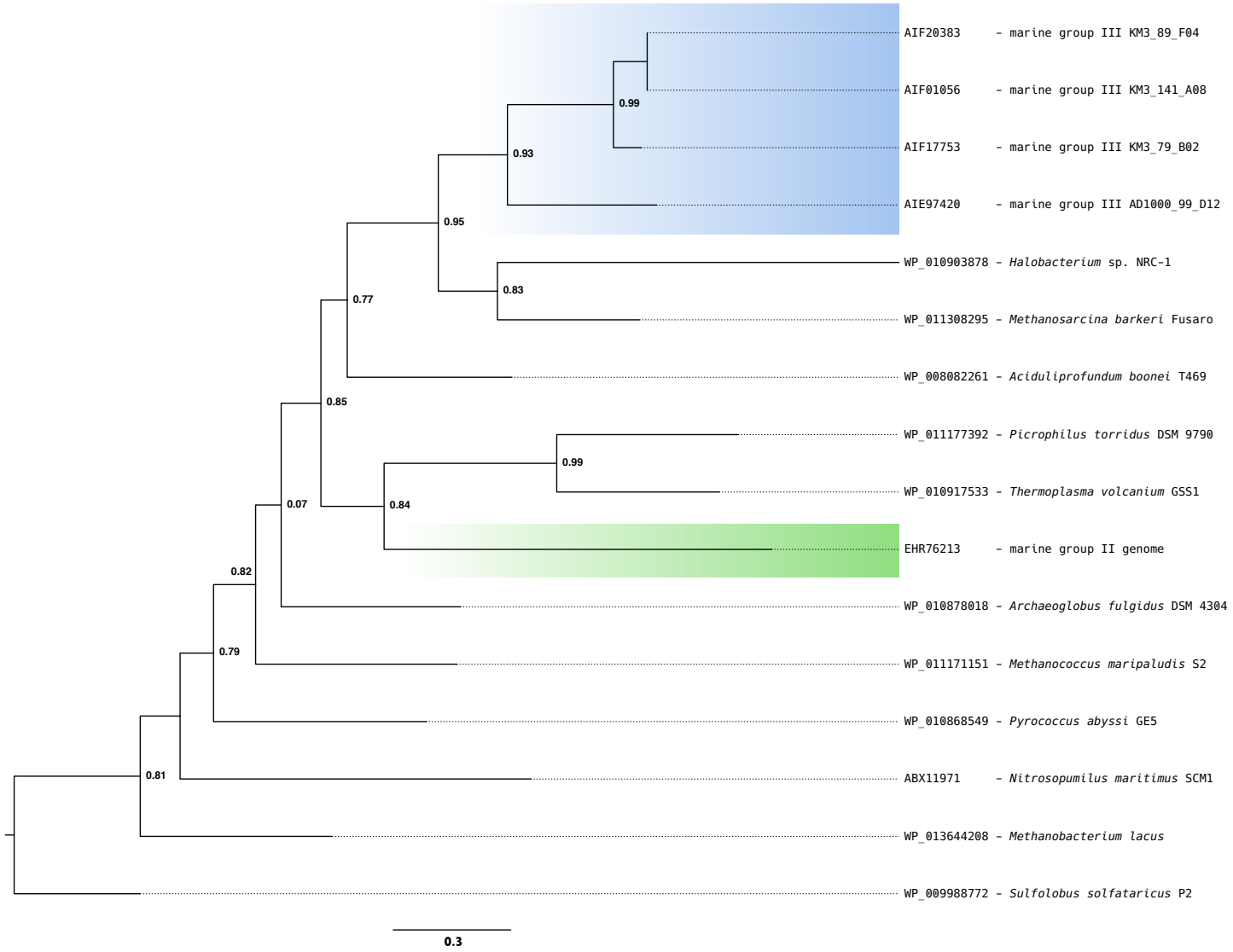

Figure A.26: Phylogeny of selected archaeal ribosomal protein S6e sequences. Highlighted are: marine group III *Euryarchaeote* sequences (blue), and marine group II *Euryarchaeote* genomic sequence (green). NCBI protein accessions are indicated. FastTree support values are shown for each branch. Branch lengths represent estimated substitutions per site with the scale shown.

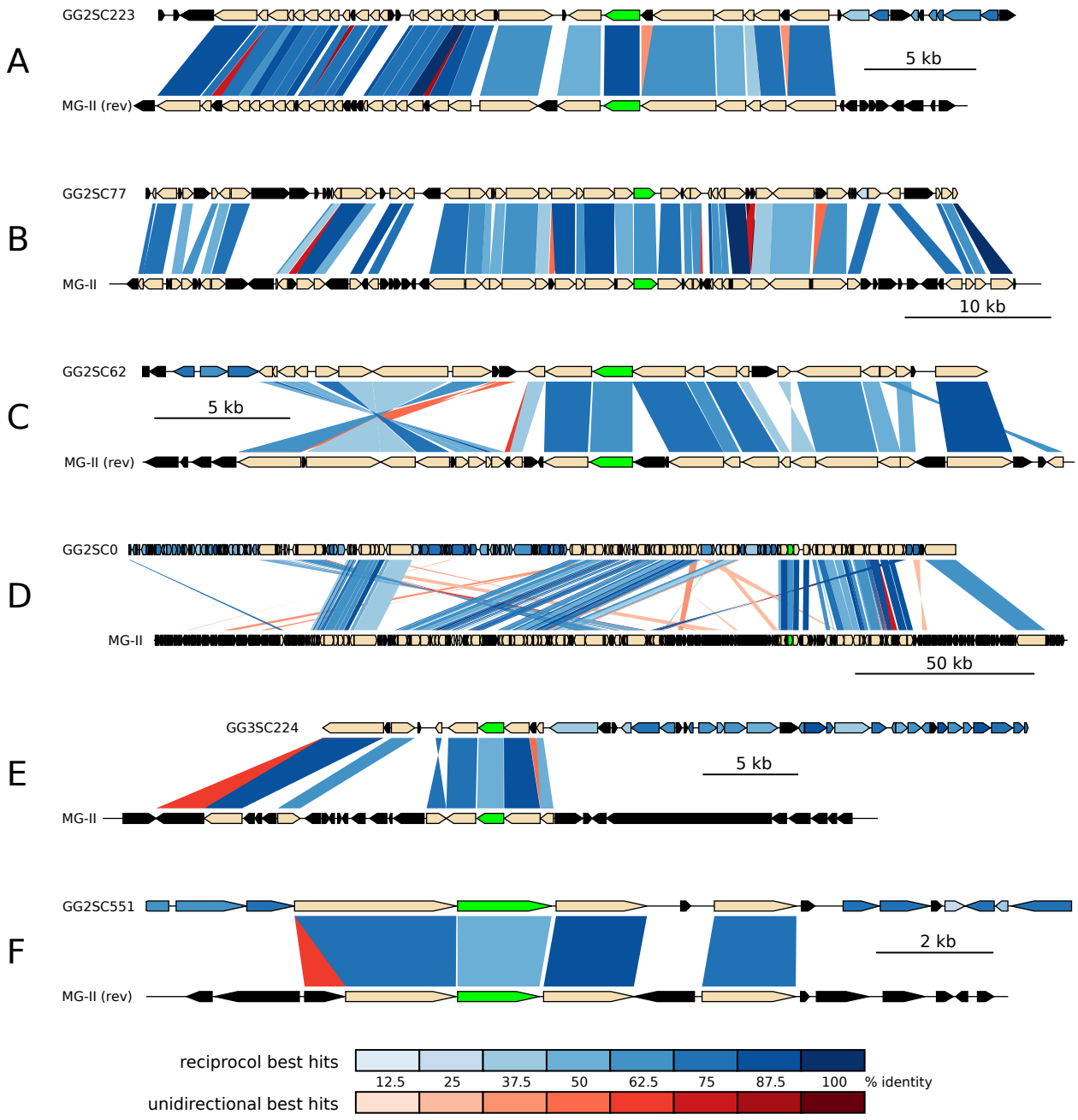

Figure A.27: Comparison of homologous proteins between the marine group II *Euryarchaeote* genome (MG-II, CM001443) and six scaffolds **(A)**–**(F)** containing thermosome subunit genes, assembled from the October and May metagenomes (see Table B.19). Homologous genes are connected by shaded regions; blue and red indicate reciprocal and unidirectional best hits, and the depth of shading indicates percent amino acid identity on the scale shown. Gene models, depicted as pointed boxes, are shaded black for genes with no blast hits, blue-shaded indicate a reciprocal best hit to an undepicted location in MG-II, green for thermosome genes and beige otherwise. MG-II sequences marked "[rev]" are reversed relative to the reference.

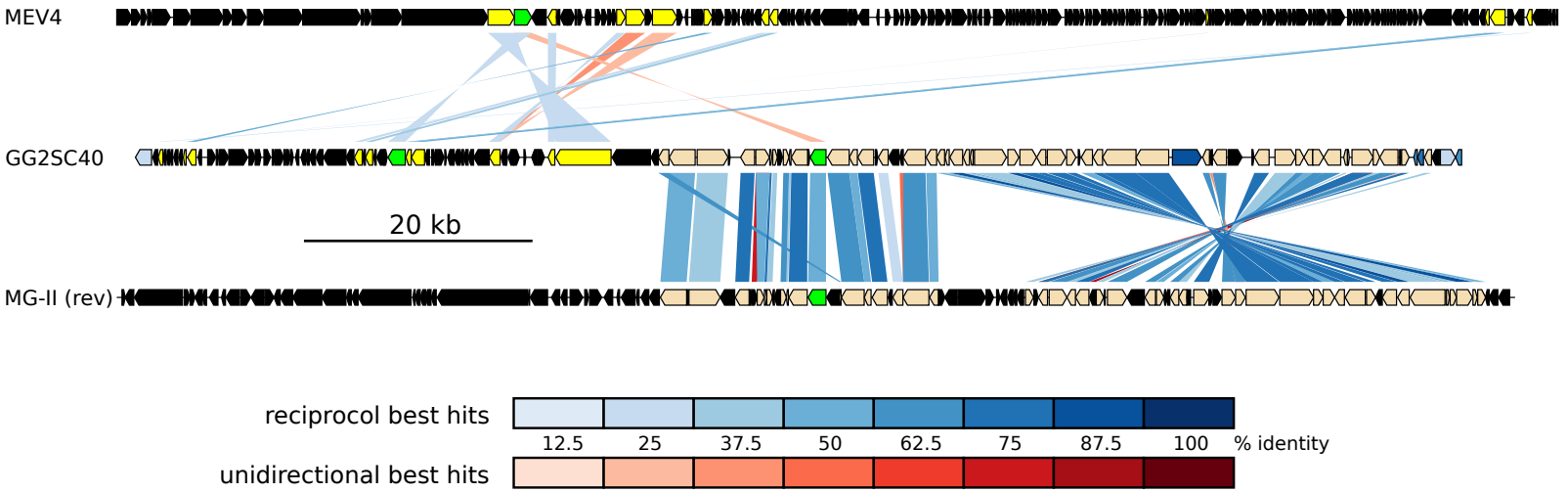

Figure A.28: Comparison of homologous proteins between metagenomic scaffold `GG2SC40` (October sample, Table B.19), the marine group II *Euryarchaeote* genome (MG-II, `CM001443`, reversed) and the group 3 virus genome MEV4. Homologous genes are connected by shaded regions; blue and red indicate reciprocal and unidirectional best hits, and the depth of shading indicates percent amino acid identity on the scale shown. Gene models (pointed boxes), are shaded black for genes with no blast hits, blue-shaded indicate a reciprocal best hit to an undepicted location in MG-II, yellow for virus genes and homologs, green for thermosome genes, and beige otherwise.

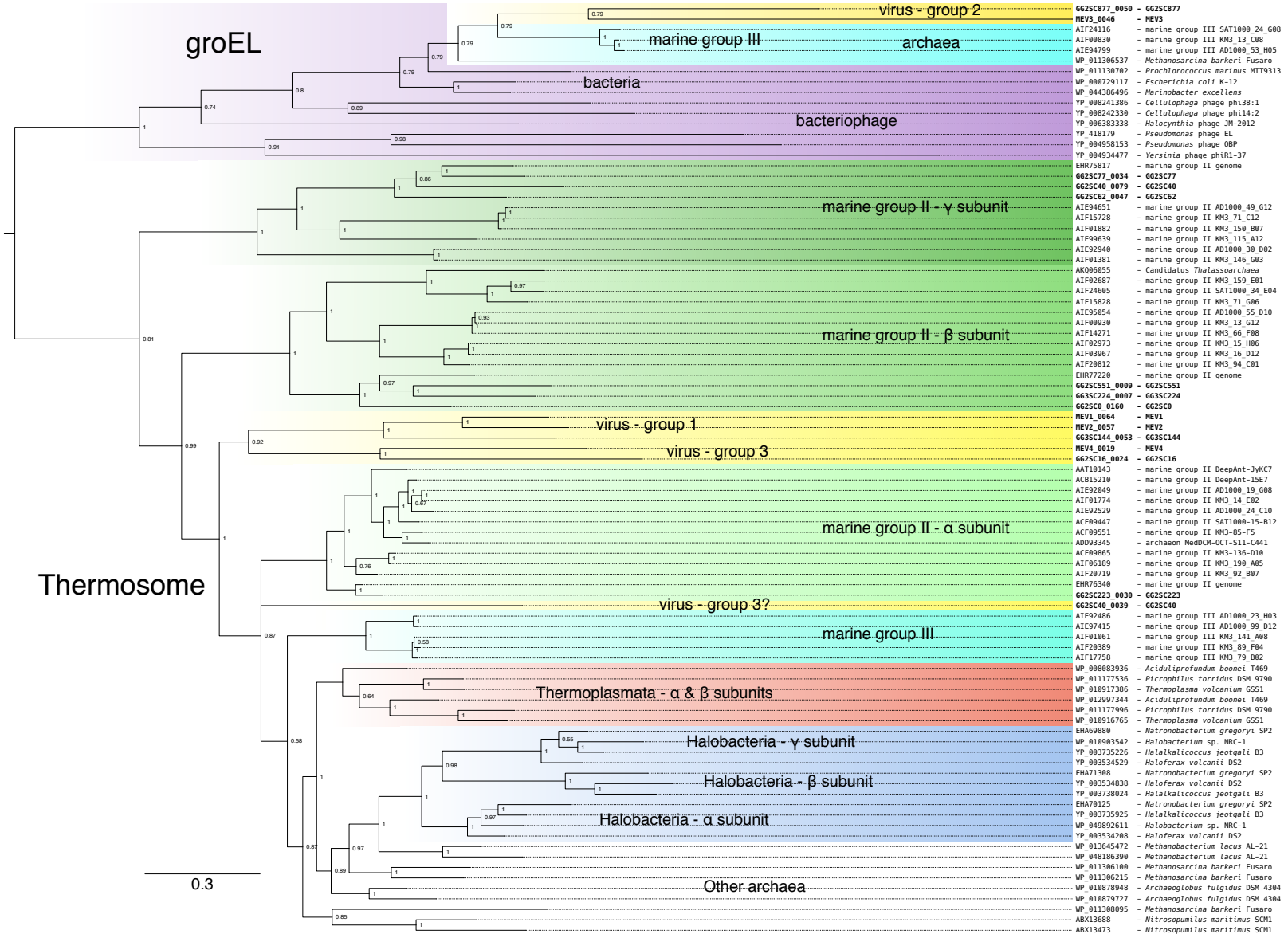

Figure A.29: Bayesian phylogeny of selected thermosome and groEL (HSP60) chaperonin proteins. Major clades are labeled taxonomically, with α, β and γ denoting paralogous subunits. Tree is rooted with the groEL subtree. NCBI Protein accessions are shown except for bold type which show locus_-ids (this study). Bayesian p-values are shown for each branch. Branch lengths represent estimated substitutions per site with the scale shown. See Figure 3.5.

Appendix B

SUPPLEMENTAL TABLES

Table B.1: Metagenome sample statistics and environmental measurements.

	October	May	
Sample collection			
Collection date	10-Oct-08	9-May-09	
Time of Day	09:00	09:30	
Depth (approximate)	1	1	meters
Location	(47.6906558, -122.404411)		degrees (lat, long)
Sample filtration			
Sample volume	91	91	liters
Prefilter	0.8	0.8	µmeters
Concentrated volume	160	150	mililiters
Nutrients and chlorophyll			
Nitrate	20.2 ± 0.2	27.4 ± 0.2	µmol / liter
Nitrite	0.76 ± 0.01	0.49 ± 0.01	µmol / liter
Ammonium	2.7 ± 0.09	1.5 ± 0.06	µmol / liter
Phosphate	2.3 ± 0.02	1.8 ± 0.01	µmol / liter
Silicate	53.5 ± 0.71	85.8 ± 1.61	µmol / liter
Chlorophyll-a	3.1	6.2	µgrams / mililiter
Clone libraries			
Bacterial 16S clones	172	-	
Archaeal 16S clones	92	-	
Cell counts - Whole Water			
Total cells (DAPI)	9.35	8.01	x 10^5 cells / mililiter
Bacteria	7.44 (80%)	6.26 (78%)	x 10^5 cells / mililiter (% of total)
SAR-11	1.76 (19%)	1.28 (16%)	x 10^5 cells / mililiter (% of total)
SAR-86	1.13 (12%)	0.81 (10%)	x 10^5 cells / mililiter (% of total)
Cell counts - Concentrate			
Total cells (DAPI)	12.8	4.66	x 10^7 cells / mililiter
Bacteria	10.2 (87%)	3.12 (67%)	x 10^7 cells / mililiter (% of total)
SAR-11	2.96 (23%)	0.67 (14%)	x 10^7 cells / mililiter (% of total)
SAR-86	1.4 (11%)	0.53 (11%)	x 10^7 cells / mililiter (% of total)

Table B.2: Summary of October 2008 metagenomic read alignment with the RefSeq (version 46) genome database. All genomes with mean read coverage >= 1-fold are listed. Read coverage as reported is competitive and shared coverage is allocated to the best alignment position(s) in the database, and coverage from reads aligning equally well to more than one position in the database is divided among those positions.

TaxID	Name	Sequence length	Mean Fold Coverage	% Covered	Genome 16S Classification (genus p-value when < 1.0)
367336	Rhodobacterales bacterium HTCC2255	2,296,803	70.5	96.0%	Bacteria; Proteobacteria; Alphaproteobacteria; Rhodobacterales; Rhodobacteraceae; Thalassobacter (0.64)
335992	Candidatus Pelagibacter ubique HTCC1062	1,308,759	35.6	95.3%	Bacteria; Proteobacteria; Alphaproteobacteria; Rickettsiales; SAR11; Pelagibacter
314261	Candidatus Pelagibacter ubique HTCC1002	1,324,008	32.9	94.1%	Bacteria; Proteobacteria; Alphaproteobacteria; Rickettsiales; SAR11; Pelagibacter
487796	Flavobacteria bacterium MS024-2A	1,905,484	5.5	90.3%	Bacteria; Bacteroidetes; Flavobacteria; Flavobacteriales; Flavobacteriaceae; NS5
136084	Roseobacter phage SIO1	39,898	5.1	75.4%	N/A
487797	Flavobacteria bacterium MS024-3C	1,515,248	2.0	84.6%	Bacteria; Bacteroidetes; Flavobacteria; Flavobacteriales; Flavobacteriaceae; NS3
439493	Candidatus Pelagibacter sp. HTCC7211	1,456,888	1.9	23.2%	Bacteria; Proteobacteria; Alphaproteobacteria; Rickettsiales; SAR11; Pelagibacter
436308	Nitrosopumilus maritimus SCM1	1,645,259	1.8	8.3%	Archaea; Thaumarchaeota; marine archaeal group I; Nitrosopumilales; Nitrosopumilaceae; Nitrosopumilus
247639	marine gamma proteobacterium HTCC2080	3,576,081	1.6	68.2%	Bacteria; Proteobacteria; Gammaproteobacteria; OMG group; OM60 clade; OM60
383631	Methylophilales bacterium HTCC2181	1,304,428	1.0	41.2%	Bacteria; Proteobacteria; Betaproteobacteria; Methylophilales; Methylophilaceae; Methylophilus (0.78)

Table B.3: Summary of May 2009 metagenomic read alignment with the RefSeq (version 46) genome database. All genomes with mean read coverage >= 1-fold are listed. Enterobacteria phage lambda DNA (yellow) was spiked into the May sample DNA as a control. Read coverage as reported is competitive and shared coverage is allocated to the best alignment position(s) in the database, and coverage from reads aligning equally well to more than one position in the database is divided among those positions.

TaxID	Name	Sequence length	Mean Fold Coverage	% Covered	Genome 16S Classification (genus p-value when < 1.0)
10710	Enterobacteria phage lambda	48,502	118.1	99.0%	N/A
367336	Rhodobacterales bacterium HTCC2255	2,296,803	56.5	96.5%	Bacteria; Proteobacteria; Alphaproteobacteria; Rhodobacterales; Rhodobacteraceae; Thalassobacter (0.64)
487796	Flavobacteria bacterium MS024-2A	1,905,484	23.0	92.9%	Bacteria; Bacteroidetes; Flavobacteria; Flavobacteriales; Flavobacteriaceae; NS5
335992	Candidatus Pelagibacter ubique HTCC1062	1,308,759	22.8	94.0%	Bacteria; Proteobacteria; Alphaproteobacteria; Rickettsiales; SAR11; Pelagibacter
314261	Candidatus Pelagibacter ubique HTCC1002	1,324,008	22.5	92.8%	Bacteria; Proteobacteria; Alphaproteobacteria; Rickettsiales; SAR11; Pelagibacter
314287	gamma proteobacterium HTCC2207	2,633,173	17.9	93.7%	Bacteria; Proteobacteria; Gammaproteobacteria; OMG group; SAR92; SAR92
487797	Flavobacteria bacterium MS024-3C	1,515,248	8.8	93.9%	Bacteria; Bacteroidetes; Flavobacteria; Flavobacteriales; Flavobacteriaceae; NS3
383631	Methylophilales bacterium HTCC2181	1,304,428	4.0	88.1%	Bacteria; Proteobacteria; Betaproteobacteria; Methylophilales; Methylophilaceae; Methylophilus (0.78)
136084	Roseobacter phage SIO1	39,898	1.7	24.4%	N/A

Table B.4: Conserved Archaeal proteins (n=31) used in the generation of the euryarchaeal phylogeny shown in Figure 2.5A.

KO	Protclust	arCOG	COG	Annotation
K06942	PRK09602	arCOG00357	COG0012	GTP-binding protein, probable translation factor
K01889	PRK04172	arCOG00410	COG0016	phenylalanyl-tRNA synthetase
K01887	CLSK2789673	arCOG00487	COG0018	Arginyl-tRNA synthetase
K02950	PRK04211	arCOG04255	COG0048	30S ribosomal protein S12
K02992	PRK04027	arCOG04254	COG0049	30S ribosomal protein S7
K02967	PRK04020	arCOG04245	COG0052	30S ribosomal protein S2
K01870	PRK06039	arCOG00807	COG0060	Isoleucyl-tRNA synthetase
K02867	PRK01143	arCOG04372	COG0080	50S ribosomal protein L11
K02863	PRK04203	arCOG04289	COG0081	50S ribosomal protein L1
K13798 or K03044/K03045	PRK08565	arCOG01762	COG0085	DNA-directed RNA polymerase subunit beta
K02906	PRK04231	arCOG04070	COG0087	50S ribosomal protein L3
K02890	PRK04223	arCOG04098	COG0091	50S ribosomal protein L22
K02982	PRK04191	arCOG04097	COG0092	30S ribosomal protein S3
K02874	PRK08571	arCOG04095	COG0093	50S ribosomal protein L14
K02931	PRK04219	arCOG04092	COG0094	50S ribosomal protein L5
K02994	PRK04034	arCOG04091	COG0096	30S ribosomal protein S8
K02933	PRK05518	arCOG04090	COG0097	50S ribosomal protein L6
K02988	PRK04044	arCOG04087	COG0098	30S ribosomal protein S5
K02952	PRK04053	arCOG01722	COG0099	30S ribosomal protein S13
K02948	PRK09607	arCOG04240	COG0100	30S ribosomal protein S11
K02871	PRK06394	arCOG04242	COG0102	50S ribosomal protein L13P
K02996	PRK00474	arCOG04243	COG0103	30S ribosomal protein S9
K02956	PRK08561	arCOG04185	COG0184	30S ribosomal protein S15
K02961	PRK08572	arCOG04096	COG0186	30S ribosomal protein S17
K02866	PRK04199	arCOG04113	COG0197	50S ribosomal protein L10/L16
K02876	PRK06419	arCOG00779	COG0200	50S ribosomal protein L15
K03076	CLSK2560501	arCOG04169	COG0201	Preprotein translocase subunit SecY
K03047	PRK00783	arCOG04241	COG0202	DNA-directed RNA polymerase subunit alpha
K02881	PRK08569	arCOG04088	COG0256	50S ribosomal protein L18
K02986	PRK04051	arCOG04239	COG0522	30S ribosomal protein S4
K07174	PRK09605	arCOG01183	COG0533	O-sialoglycoprotein endopeptidase

Table B.5: Genome Locus IDs of proteins used in the generation of the euryarchaeal phylogeny shown in Figure 2.5A.

KO	MG-II	*A. boonei*	*T. volcanum*	*P. torridus*	*M. barkeri*	*A. fulgidus*	*P. abyssi*	M. sp. AL-21	H. sp. NRC-1	*N. maritimus*	*M. maripaludis*
K06942	MG2_0309	Aboo_0538	TVN1009	PTO0852	Mbar_A1073	AF1364	PAB1357	Metbo_0024	VNG0504G	Nmar_1534	MMP1122
K01889	MG2_1359	Aboo_0186	TVN1037	PTO0125	Mbar_A1375	AF1955	PAB2426	Metbo_1081	VNG2504G	Nmar_1489	MMP1496
K01887	MG2_0811	Aboo_0562	TVN1320	PTO0603	Mbar_A1008	AF0894	PAB0469	Metbo_0089	VNG6312G	Nmar_0686	MMP1026
K02950	MG2_1069	Aboo_0070	TVN0163	PTO0853	Mbar_A3688	AF1892	PAB0427	Metbo_2392	VNG2658G	Nmar_0354	MMP1367
K02992	MG2_1068	Aboo_0071	TVN0162	PTO0854	Mbar_A3687	AF1893	PAB0428	Metbo_2391	VNG2657G	Nmar_0355	MMP1368
K02967	MG2_452	Aboo_1302	TVN0399	PTO0517	Mbar_A1423	AF1133	PAB0368	Metbo_0816	VNG1143G	Nmar_0316	MMP0667
K01870	MG2_1459	Aboo_0154	TVN0829	PTO0411	Mbar_A3613	AF0633	PAB0616	Metbo_0154	VNG2190G	Nmar_1120	MMP1474
K02867	MG2_0382	Aboo_0863	TVN0423	PTO0437	Mbar_A0615	AF0538	PAB2353	Metbo_2250	VNG1108G	Nmar_0385	MMP1433
K02863	MG2_0383	Aboo_0862	TVN0422	PTO0438	Mbar_A0614	AF1490	PAB1166	Metbo_2251	VNG1105G	Nmar_0382	MMP0260
K13798	MG2_0882	Aboo_1091	TVN1182	PTO0259	-	-	PAB0423	-	-	Nmar_0347	-
K03044	-	-	-	-	Mbar_A3693	AF1887	-	Metbo_2397	VNG2665G	-	MMP1362
K03045	-	-	-	-	Mbar_A3694	AF1886	-	Metbo_2398	VNG2666G	-	MMP1361
K02906	MG2_0756	Aboo_1521	TVN0324	PTO0640	Mbar_A0110	AF1925	PAB2120	Metbo_0774	VNG1689G	Nmar_0809	MMP1543
K02890	MG2_0761	Aboo_1516	TVN0329	PTO0645	Mbar_A0105	AF1920	PAB2396	Metbo_0779	VNG1695G	Nmar_0805	MMP1403
K02982	MG2_0762	Aboo_1515	TVN0330	PTO0646	Mbar_A0104	AF1919	PAB2125	Metbo_0780	VNG1697G	Nmar_0804	MMP1404
K02874	MG2_0767	Aboo_1511	TVN0335	PTO0650	Mbar_A0100	AF1915	PAB2436	Metbo_0785	VNG1701G	Nmar_0800	MMP1409
K02931	MG2_0770	Aboo_1508	TVN0338	PTO0653	Mbar_A0097	AF1912	PAB2130	Metbo_0788	VNG1705G	Nmar_0797	MMP1412
K02994	MG2_0772	Aboo_1506	TVN0340	PTO0655	Mbar_A0095	AF1910	PAB2131	Metbo_0790	VNG1707G	Nmar_0795	MMP1414
K02933	MG2_0773	Aboo_1505	TVN0341	PTO0656	Mbar_A0094	AF1909	PAB2132	Metbo_0791	VNG1709G	Nmar_0794	MMP1415
K02988	MG2_0777	Aboo_1501	TVN0345	PTO0660	Mbar_A0090	AF1905	PAB2136	Metbo_0795	VNG1715G	Nmar_0399	MMP1419
K02952	MG2_0134	Aboo_1396	TVN0562	PTO1219	Mbar_A0078	AF2285	PAB0360	Metbo_0805	VNG1132G	Nmar_0325	MMP1319
K02948	MG2_0132	Aboo_1398	TVN0564	PTO1221	Mbar_A0076	AF2283	PAB0362	Metbo_0807	VNG1134G	Nmar_1450	MMP1321
K02871	MG2_0262	Aboo_1475	TVN1135	PTO0323	Mbar_A1427	AF1128	PAB0365	Metbo_0810	VNG1138G	Nmar_0426	MMP1324
K02996	MG2_0263	Aboo_1476	TVN1136	PTO0324	Mbar_A1426	AF1129	PAB0366	Metbo_0811	VNG1139Gm	Nmar_0425	MMP1325
K02956	MG2_1260	Aboo_1376	TVN1208	PTO0244	Mbar_A1873	AF0801	PAB0033	Metbo_0103	VNG0790G	Nmar_1508	MMP1579
K02961	MG2_0766	Aboo_1512	TVN0334	PTO0649	Mbar_A0101	AF1916	PAB2127	Metbo_0784	VNG1700G	Nmar_0801	MMP1408
K02866	MG2_0678	Aboo_0067	TVN0539	PTO0715	Mbar_A1371	AF1339	PAB1444	Metbo_0379	VNG0099G	Nmar_0449	MMP1289
K02876	MG2_0779	Aboo_1499	TVN0347	PTO0662	Mbar_A0088	AF1903	PAB2138	Metbo_0797	VNG1718G	Nmar_0401	MMP1421
K03076	MG2_0780	Aboo_1498	TVN0348	PTO0663	Mbar_A0087	AF1902	PAB2139	Metbo_0798	VNG1719G	Nmar_0402	MMP1422
K03047	MG2_0131	Aboo_1399	TVN0565	PTO1222	Mbar_A0075	AF2282	PAB2410	Metbo_0808	VNG1136G	Nmar_0428	MMP1322
K02881	MG2_0776	Aboo_1502	TVN0344	PTO0659	Mbar_A0091	AF1906	PAB2135	Metbo_0794	VNG1714G	Nmar_0398	MMP1418
K02986	MG2_0133	Aboo_1397	TVN0563	PTO1220	Mbar_A0077	AF2284	PAB0361	Metbo_0806	VNG1133G	Nmar_0324	MMP1320
K07174	MG2_1145	Aboo_0655	TVN1276	PTO0374	Mbar_A0279	AF0665	PAB1047	Metbo_0101	VNG2045G	Nmar_1531	MMP0415

Table B.6: Ribosomal proteins found in the marine group II *Euryarchaeote* and *Aciduliprofundum boonei* (NC_013926) genomes. Blue and green shading alternates and denotes groups of syntenic genes.

MG-II Locus ID	*A. boonei* accession	Name
MG2_0132	YP_003483767	ribosomal protein S11P
MG2_0133	YP_003483766	ribosomal protein S4
MG2_0134	YP_003483765	ribosomal protein S13P
MG2_0175	YP_003483627	ribosomal protein LX
MG2_0261	YP_003483842	ribosomal protein L15
MG2_0262	YP_003483843	ribosomal protein L13
MG2_0263	YP_003483844	ribosomal protein S9P
MG2_0362	YP_003483136	ribosomal protein L21e
MG2_0382	YP_003483234	ribosomal protein L11
MG2_0383	YP_003483233	ribosomal protein L1
MG2_0384	YP_003483232	ribosomal protein L10
MG2_0385	YP_003483231	ribosomal protein L12
MG2_0423	YP_003483497	ribosomal protein L15e
MG2_0452	YP_003483672	ribosomal protein S2
MG2_0625	YP_003483173	ribosomal protein S6e
MG2_0678	YP_003482441	ribosomal protein L10e
MG2_0687	YP_003483524	ribosomal protein S27E
MG2_0688	YP_003483525	ribosomal protein L44E
MG2_0691	YP_003483759	ribosomal protein L37e
MG2_0756	YP_003483889	ribosomal protein L3
MG2_0757	YP_003483888	ribosomal protein L4P
MG2_0758	YP_003483887	ribosomal protein L23
MG2_0759	YP_003483886	ribosomal protein L2
MG2_0760	YP_003483885	ribosomal protein S19
MG2_0761	YP_003483884	ribosomal protein L22
MG2_0762	YP_003483883	ribosomal protein S3
MG2_0763	YP_003483882	ribosomal protein L29
MG2_0766	YP_003483880	ribosomal protein S17P
MG2_0767	YP_003483879	ribosomal protein L14P
MG2_0768	YP_003483878	ribosomal protein L24
MG2_0769	YP_003483877	ribosomal protein S4e
MG2_0770	YP_003483876	ribosomal protein L5
MG2_0771	YP_003483875	ribosomal protein S14
MG2_0772	YP_003483874	ribosomal protein S8
MG2_0773	YP_003483873	ribosomal protein L6P
MG2_0774	YP_003483872	ribosomal protein L32e
MG2_0775	YP_003483871	ribosomal protein L19e
MG2_0776	YP_003483870	ribosomal protein L18P
MG2_0777	YP_003483869	ribosomal protein S5
MG2_0778	YP_003483868	ribosomal protein L30P
MG2_0779	YP_003483867	ribosomal protein L15
MG2_0885	YP_003483459	ribosomal protein L30e
MG2_0923	YP_003483006	ribosomal protein S3Ae
MG2_1068	YP_003482445	ribosomal protein S7
MG2_1069	YP_003482444	ribosomal protein S12
MG2_1079	YP_003482554	ribosomal protein L37a
MG2_1161	YP_003483616	ribosomal protein S17e
MG2_1162	YP_003482401	ribosomal protein L31e
MG2_1163	YP_003482400	ribosomal protein L39e
MG2_1165	YP_003482398	ribosomal protein S19e
MG2_1232	YP_003482937	ribosomal protein S27a
MG2_1233	YP_003482938	ribosomal protein S24e
MG2_1260	YP_003483746	ribosomal protein S15P
MG2_1387	YP_003483682	ribosomal protein L40e
MG2_1704	YP_003483606	ribosomal protein L7Ae
MG2_1705	YP_003483607	ribosomal protein S28e
MG2_1706	YP_003483608	ribosomal protein L24E
MG2_1729	YP_003482448	ribosomal protein S10
MG2_1736	YP_003483781	ribosomal protein S8e

Table B.7: Putative peptidase genes and non-peptidase homologs identified in the marine group II *Euryarchaeote* genome. MEGAN classifications indicate the most specific taxonomic group with which the protein identified. Archaeal classifications are shaded for visibility.

Locus ID	Length (nt)	MEROPS id	E-value	MEGAN class.	Product
MG2_0002	1013	MER001728	6.60E-21	Euryarchaeota	methionyl aminopeptidase 2 (M24A subfamily)
MG2_0059	3407	MER000383	3.70E-50	Chroococcales	proprotein convertase, kexin type 2, (S8B subfamily)
MG2_0167	1835	MER161420	6.70E-104	Actinomycetales	X-Pro dipeptidyl-peptidase (S15 family)
MG2_0176	3128	MER085659	1.10E-25	Deltaproteobacteria	KP-43 subtilisin peptidase (S8A subfamily)
MG2_0201	1745	MER001284	2.30E-05	Bacteria	Zn-dependent peptidase (M28E subfamily)
MG2_0207	1586	MER002161	2.60E-10	Bacteroidetes	Zn-dependent peptidase (M28A subfamily)
MG2_0265	4724	MER133814	4.80E-06	none	putative subtilisin peptidase (S8A subfamily)
MG2_0275	1184	MER180647	1.20E-08	Halobacteriaceae	conserved cell surface protein similar to PKD/Chitinase/Collegenase domain proteins (putative S8A and M9 family peptidases)
MG2_0303	965	MER000431	4.00E-72	Cellular organisms	prolyl aminopeptidase (S33 family)
MG2_0348	1274	MER002038	4.00E-09	Bacteria	putative X-Pro metallopeptidase (M24B subfamily)
MG2_0389	3815	MER133814	1.80E-05	Bacteria	putative subtilisin peptidase (S8A subfamily)
MG2_0401	2123	MER001212	3.20E-16	Bacteria	metallocarboxypeptidase T (M14A subfamily)
MG2_0429	4538	MER016991	3.70E-74	cellular organism	KP-43 subtilisin peptidase (S8A subfamily)
MG2_0447	1805	MER133814	6.40E-06	cellular organisms	putative peptidase (S8A subfamily)
MG2_0492	1958	MER161420	2.40E-110	Bacteria	X-Pro dipeptidyl-peptidase (S15 family)
MG2_0512	1496	MER001186	1.10E-116	cellular organism	Zn-dependent carboxypeptidase Taq (M32 family)
MG2_0565	1367	MER058960	7.30E-05	none	putative subtilisin peptidase (S8A subfamily)
MG2_0567	2009	MER019280	6.10E-56	cellular organisms	tengconlysin-like subtilisin peptidase (S8A subfamily)
MG2_0636	1835	MER073084	3.00E-50	Bacteria	BcepAP-like glycyl aminopeptidase (M61 family)
MG2_0645	1229	MER100667	3.40E-11	Chloroflexi	predicted aminopeptidase (M28A subfamily)
MG2_0646	4208	MER180647	6.10E-09	Flavobacteriales	conserved putative peptidase (M30 and S8A family domains)
MG2_0652	1448	MER001205	9.60E-05	cellular organisms	putative metallocarboxypeptidase (M14 family), similar to Succinylglutamate desuccinylase (ASTE)/aspartoacylase (ASPA)
MG2_0659	1436	MER145055	3.20E-78	Euryarchaeota	peptidase U62, putative microcin-like modulator of DNA gyrase
MG2_0660	1403	MER051962	2.60E-26	Euryarchaeota	peptidase U62, putative microcin-like modulator of DNA gyrase
MG2_0747	2195	MER004123	5.50E-30	cellular organisms	subtilisin peptidase with homology to subA cytotoxin (S8A subfamily)
MG2_0818	1784	MER005239	1.50E-12	cellular organisms	Zn-dependent peptidase (M28B subfamily) with an SAP DNA binding domain
MG2_0905	5318	MER133814	4.80E-06	Bacteria	conserved putative beta-lytic metallopeptidase (M23B subfamily)
MG2_0906	6119	MER133814	9.20E-07	Bacteria	conserved putative peptidase (U69 and S8A family homology)
MG2_0909	3326	MER145093	4.10E-07	Bacteria	conserved putative peptidase (U69 and S8A family homology)
MG2_0910	3695	MER134526	2.30E-09	Bacteria	conserved putative peptidase (U69 and S8A family homology)
MG2_0955	626	MER000548	8.60E-47	Euryarchaeota	proteasome, beta component (T1A subfamily)
MG2_1084	722	MER017631	2.40E-59	Euryarchaeota	proteasome, alpha component (T1 family)
MG2_1145	569	MER006226	2.40E-38	Euryarchaeota	bifunctional UGMP fusion protein (Bud32 protein kinase / M22 family, Kae1 putative peptidase)
MG2_1183	2120	MER075049	5.90E-19	cellular organisms	Immune inhibitor A-like metalloendopeptidase (M6 family)
MG2_1200	1412	MER001244	1.20E-67	Bacteria	X-Pro aminopeptidase (M24B subfamily)
MG2_1217	2582	MER145048	3.80E-22	Bacteria	bacillopeptidase F-like subtilisin (S8A subfamily)
MG2_1226	1253	MER019280	1.10E-34	cellular organisms	subtilisin peptidase with homology to subA cytotoxin (S8A subfamily)
MG2_1245	4985	MER051661	2.40E-47	cellular organisms	KP-43 subtilisin peptidase (S8A subfamily)
MG2_1281	1220	MER019280	4.10E-20	cellular organisms	subtilisin peptidase with homology to subA cytotoxin (S8A subfamily)
MG2_1361	1277	MER161420	1.50E-118	Bacteria	putative X-Pro dipeptidyl-peptidase (S15 family)
MG2_1362	521	MER161420	3.70E-37	none	putative X-Pro dipeptidyl-peptidase (S15 family)
MG2_1374	746	MER173648	1.50E-10	Bacteria	putative prolyl oligopeptidase (S9 family)
MG2_1382	2078	MER059659	2.70E-71	Euryarchaeota	Lon-B peptidase, ATP-dependent (S16 family)
MG2_1463	1808	MER161420	9.20E-127	Actinomycetales	X-Pro dipeptidyl-peptidase (S15 family)
MG2_1486	2783	MER040490	5.60E-33	cellular organisms	subtilisin peptidase with homology to subA cytotoxin (S8A subfamily)
MG2_1487	989	MER017194	9.40E-08	Bacteria	Rhomboid family protein, possible intramembrane serine protease (S54 family)
MG2_1493	635	MER180577	5.10E-10	Bacteria	putative lysostaphin peptidase (M23B subfamily)
MG2_1525	5642	MER051661	8.20E-36	cellular organisms	KP-43 subtilisin peptidase (S8A subfamily) with thrombospondin type 3 repeats
MG2_1528	1490	MER001236	2.00E-69	cellular organisms	multifunctional PepA aminopeptidase (M17 family)
MG2_1529	1346	MER161667	3.00E-15	cellular organisms	putative carboxypeptidase T (M14A subfamily)
MG2_1636	1472	MER001288	1.10E-06	Bacteria	putative Zn-dependent exopeptidase (M28E subfamily)
MG2_1667	1553	MER058052	2.00E-06	Euryarchaeota	putative zinc metalloprotease (M50B subfamily)

Table B.8: Putative non-peptidase homologs identified in the marine group II *Euryarchaeote* genome. MEGAN classifications indicate the most specific taxonomic group with which the protein identified. Archaeal classifications are shaded for visibility.

Locus ID	Length (nt)	MEROPS ID	E-value	MEGAN class.	Product
MG2_0003	608	MER151562	3.60E-20	cellular organisms	putative translation factor SUA5, (M22 family non-peptidase homolog)
MG2_0123	524	MER090044	1.60E-24	Archaea	GMP synthase / gamma-glutamyl hydrolase, (C26 family unassigned peptidase)
MG2_0184	851	MER042827	2.80E-08	Bacteria	phosphoribosylformylglycinamidine synthase (C56 family non-peptidase homolog)
MG2_0311	2195	MER134379	4.70E-35	Bacteria	Anthranilate/para-aminobenzoate synthase component I (C26 family unassigned peptidase homolog)
MG2_0343	1259	MER037714	9.90E-12	Euryarchaeota	Cytosine deaminase or related metal-dependent hydrolase (M38 family non-peptidase homolog)
MG2_0387	6569	MER133814	6.00E-05	cellular organisms	regulator of chromosome condensation RCC1 with PRK-like repeat, fucose specific lectin
MG2_0514	1418	MER005767	6.40E-18	Bacteria	dihydroorotase (M38 family non-peptidase homolog)
MG2_0541	2114	MER077132	1.10E-16	Eukaryota	ABC-type phosphate/phosphonate transport system, periplasmic component (S60 family non-peptidase homolog)
MG2_0692	1538	MER003314	7.30E-32	Bacteria	amidophosphoribosyltransferase (C44 family)
MG2_0799	1334	MER060647	4.20E-61	cellular organisms	carbamoyl-phosphate synthase small subunit (C26 family non-peptidase homolog)
MG2_0857	1766	MER033254	2.20E-16	cellular organisms	asparagine synthase, glutamine-hydrolysing (C44 family non-peptidase homolog)
MG2_1094	3665	MER180647	1.80E-06	Bacteria	putative C-type lectin with I63 family peptidase inhibitor-like homology
MG2_1144	1358	MER033186	1.10E-48	cellular organisms	Imidazolonepropionase or related amidohydrolase (M38 family non-peptidase homolog)
MG2_1484	1820	MER003327	5.30E-55	Bacteria	glucosamine–fructose-6-phosphate aminotransferase, isomerizing (C44 family non-peptidase homolog)

Table B.9: Putative lipid metabolism genes in the marine group II *Euryarchaeote* genome. MEGAN classifications indicate the most specific taxonomic group with which the protein identified. Archaeal classifications are shaded for visibility.

Locus ID	Gene/EC number	KO/ProtClust	MEGAN class.	Annotation
Fatty acid synthesis				
MG2_1652	6.3.4.14 / 6.4.1.2	K01961	cellular organisms	acetyl-CoA carboxylase, biotin carboxylase subunit
MG2_0944-5	FabH / 2.3.1.180	K00648	Actinomycetales	3-oxoacyl-[acyl-carrier-protein] synthase III
MG2_0114	FabG / 1.1.1.100	K00059	cellular organisms	3-ketoacyl-(acyl-carrier-protein) reductase
MG2_1373	FabG / 1.1.1.100	K00059	Bacteria	3-ketoacyl-(acyl-carrier-protein) reductase
MG2_1640	FabI / 1.3.1.9	K00208	Bacteria	enoyl-[acyl-carrier protein] reductase I
MG2_0458	4.2.1.17	K00074	Firmicutes	bifunctional 3-hydroxyacyl-CoA dehydrogenase / enoyl-CoA hydratase / isomerase
MG2_1647	3.1.2.20	K01076	cellular organisms	putative acyl-CoA hydrolase
PUFA synthesis				
MG2_1639	2.3.1.94	K10817	Chloroflexi	Polyketide synthase, beta-ketoacyl synthase domain
MG2_1647	3.1.2.20	K01076	cellular organisms	putative acyl-CoA hydrolase
MG2_1638	2.7.8.-	K06133	Alphaproteobacteria	putative 4'-phosphopantetheinyl transferase
MG2_0653		CLSK946835	Myxococcales	thioesterase superfamily protein
MG2_0738	1.14.19.1	K00507	Proteobacteria	steroyl-CoA desaturase (delta-9 fatty acid desaturase)
MG2_1035	1.14.21.6	K00227	Bacteria	putative lathosterol oxidase, fatty acid hydroxylase superfamily
Ether-linked lipids				
MG2_0160		CLSK2468350	Bacteroidetes	putative geranylgeranyl reductase
MG2_1195		CLSK227774	Euryarchaeota	putative geranylgeranyl reductase
MG2_1005	2.5.1.-	PRK09573	Euryarchaeota	4-hydroxybenzoate octaprenyltransferase
MG2_0180		PRK04169	Euryarchaeota	putative geranylgeranylglyceryl phosphate synthase
MG2_0580	1.4.1.-	K11646	cellular organisms	3-dehydroquinate synthase II
MG2_1065	2.5.1.1 / 2.5.1.10	K13787	cellular organisms	geranylgeranyl diphosphate synthase
MG2_0670	2.5.1.-	K02523	cellular organisms	octaprenyl diphosphate synthase / Geranylgeranylpyrophosphate synthase
Glycerophospholipids				
MG2_1620	2.7.1.30???	CLSK865122	cellular organisms	Predicted Inorganic polyphosphate / ATP-NAD kinase
MG2_1285	2.7.7.41	CLSK2789688	Euryarchaeota	CDP-diglyceride synthetase
MG2_0037	2.7.1.107	K04718	Chlorophyta	sphingosine kinase, diacyl glycerol kinase
MG2_1540	3.1.3.4	CLSK2770155	cellular organisms	putative membrane-associated phospholipid phosphatase
MG2_1125	3.1.1.23	CLSK951691	no hit > 35 bits	alpha/beta hydrolase fold protein
MG2_0104	2.3.1.15 / 2.3.1.42	K13507	cellular organisms	putative Lysophospholipid acyltransferase (LPLAT)
MG2_0568	2.3.1.15 / 2.3.1.42	K00655	cellular organisms	putative Phospholipid / glycerol acyltransferase
MG2_0929	2.3.1.51	K00655	Bacteria	1-acyl-sn-glycerol-3-phosphate acyltransferase or related
MG2_0546	1.1.1.94	K00057	Bacteria	NAD(P)H-dependent glycerol-3-phosphate dehydrogenase
MG2_1675		CLSK903188	Plesiocystis pacifica	putative integral membrane protein with PLC-like phosphodiesterase
MG2_1349		PF03009 pfam	Clostridiales	putative Glycerophosphoryl diester phosphodiesterase
MG2_1350		CLSK752696	cellular organisms	putative Membrane-associated phospholipid phosphatase
MG2_1352		CLSK2835513	Bacteria	Metal-dependent phosphoesterase (PHP family)
Fatty acid metabolism				
MG2_1503	6.2.1.3	K01897	Myxococcales	long-chain acyl-CoA synthetase
MG2_0560	1.3.99.3	K00249	cellular organisms	Acyl-CoA dehydrogenase
MG2_0630	1.3.99.3	K00249	Bacteria	Acyl-CoA dehydrogenase
MG2_1576	4.2.1.17	K01692	Bacteria	Enoyl-CoA hydratase / isomerase
MG2_0458	4.2.1.17 / 1.1.1.211	K00074	Firmicutes	bifunctional 3-hydroxyacyl-CoA dehydrogenase / enoyl-CoA hydratase / isomerase
MG2_1061	2.3.1.16	K00632	cellular organisms	acetyl-CoA acetyltransferase Protein
MG2_0518	2.3.1.9	K00626	Bacteria	acetyl-CoA C-acetyltransferase Protein
MG2_0786	2.3.1.9	K00626	cellular organisms	acetyl-CoA C-acetyltransferase Protein
MG2_0702	1.3.99.7	K00252	cellular organisms	Glutaryl-CoA dehydrogenase
Alkane degradation				
MG2_0476	1.14.15.3	K00496	Bacteria	alkane 1-monooxygenase
MG2_0934	1.1.1.1	K00001	cellular organisms	alcohol dehydrogenase-like protein
MG2_1561	1.2.1.3	K00128	Halobacteriaceae	aldehyde dehydrogenase
MG2_1600	1.2.1.3	K00128	Bacteria	aldehyde dehydrogenase

Table B.10: Results of pairwise reciprocal blast analyses between proteins coded in seven archaeal genomes. Counts of homologous proteins shared between each pair of genomesl as determined by reciprocal best hits (RBHs)l are shown above the diagonal. Below the diagonal is the fraction of proteins shared between each pair of genomesl calculated as percentage of shared homologs for the genome with the smallest number of total proteins. Percentages are shaded by decile for visibility.

	MG-II	*A. boonei*	*P. torridus*	*T. volcanium*	*A. fulgidus*	*P. abyssi*	*N. maritimus*
MG-II		506	472	476	505	468	440
A. boonei	33%		655	668	702	770	487
P. torridus	30%	42%		1047	620	578	535
T. volcanium	30%	43%	68%		636	599	512
A. fulgidus	28%	45%	40%	41%		819	583
P. abyssi	26%	50%	37%	38%	43%		538
N. maritimus	25%	31%	34%	33%	32%	30%	

Table B.11: Assignment of non-homologous proteins among three comparison genomes to NCBI taxonomic groups. The table background is shaded by domain level classification (corresponding with the color scheme of Figure A.18). Top sub-domain groups are shown underlined and taxonomic groups within them that received classified proteins are shown belowl within the same box. Counts represent the number of proteins classified by MEGAN at that taxonomic level. Sums are calculated for each domain and sub-domain. Totaled sums in bold type correspond with the counts shown on Figure 2.5C. The genomes used are: marine group II *Euryarchaeote*, *Aciduliprofundum boonei* (NC_013926) and *Nitrosopumilus maritimus* (NC_010085). See Methods for details of the analysis.

	Mar. Group-II		*A. boonei*		*N. maritimus*		**Total**	
	Count	Sum	Count	Sum	Count	Sum	**Count**	**Sum**
root	4	1126	1	848	2	1161	7	**3135**
cellular organisms	128	544	140	625	186	647	454	**1816**
Bacteria	180	332	77	119	177	262	434	**713**
Actinobacteria	0	14	0	0	0	6	0	**20**
Actinobacteria (class)	0		0		6		6	
Actinobacteridae (subclass)	1		0		0		1	
Actinomycetales (order)	13		0		0		13	
Bacteroidetes/Chlorobi group	0	17	5	5	0	0	5	**22**
Bacteroidetes (phylum)	11		0		0		11	
Flavobacteria (class)	6		0		0		6	
Chloroflexi	6	11	0	0	14	14	20	**25**
Chloroflexi (class)	5		0		0		5	
Cyanobacteria	4	9	0	0	7	7	11	**16**
Chroococcales (order)	5		0		0		5	
Firmicutes	9	15	2	17	6	15	17	**47**
Bacillales (order)	0		0		3		3	
Bacillus (genus)	0		0		6		6	
Clostridia (class)	6		7		0		13	
Clostridiales (order)	0		8		0		8	
Planctomycetes	0	9	0	0	0	0	0	**9**
Planctomycetaceae (family)	4		0		0		4	
Planctomyces (genus)	5		0		0		5	
Proteobacteria	18	77	3	14	17	43	38	**134**
Alphaproteobacteria (class)	11		0		7		18	
Betaproteobacteria (class)	6		0		8		14	
delta/epsilon subdiv. (subphylum)	0		2		0		2	
Deltaproteobacteria (class)	11		9		0		20	
Myxococcales (order)	6		0		0		6	
Gammaproteobacteria (class)	25		0		11		36	
Thermotogae	0	0	0	6	0	0	0	**6**
Thermotogales (order)	0		1		0		1	
Thermotogaceae (family)	0		5		0		5	
Archaea	6	64	51	366	40	187	97	**617**
Crenarchaeota	0	5	0	22	0	44	0	**71**
Thermoprotei (class)	5		2		17		24	
Desulfurococcales (order)	0		3		0		3	
Desulfurococcaceae (family)	0		5		8		13	
Sulfolobaceae (family)	0		5		4		9	
Sulfolobus (genus)	0		0		6		6	
Thermoproteales (order)	0		7		3		10	
Thermoproteaceae (family)	0		0		6		6	
Euryarchaeota	32	53	83	288	47	94	162	**435**
Archaeoglobaceae (family)	0		8		6		14	
Archaeoglobus (genus)	0		8		0		8	
Halobacteriaceae (family)	15		5		10		30	
Methanobacteriales (order)	0		1		0		1	
Methanobacteriaceae (family)	0		7		0		7	
Methanococcales (order)	0		7		0		7	
Methanomicrobia (class)	6		12		11		29	
Methanomicrobiales (order)	0		7		0		7	
Methanosarcinales (order)	0		0		2		2	
Methanosaeta thermophila	0		6		0		6	
Methanosarcinaceae (family)	0		11		2		13	
Methanosarcina (genus)	0		0		5		5	
Thermococcaceae (family)	0		79		6		85	
Thermococcus (genus)	0		54		0		54	
Thermoplasmatales (order)	0		0		5		5	
Korarchaeota	0	0	0	5	0	9	0	**14**
Candidatus Korarchaeum cryptofilum	0		5		9		14	
Eukaryota	12	20	0	0	6	12	18	**32**
Fungi/Metazoa group	0	0	0	0	1	6	1	**6**
Metazoa (kingdom)	0		0		5		5	
Viridiplantae	2	8	0	0	0	0	2	**8**
Chlorophyta (phylum)	6		0		0		6	
no hits	578	578	222	222	512	512	1312	**1312**

Table B.12: Metagenome *de novo* assembly statistics for October and May samples for each k-mer length used.

K-mer	Sample	Contigs	Total sequence	N50
23	October	295571	102460533	593
23	May	250311	101624229	772
25	October	250445	96114036	663
25	May	213551	96644258	873
27	October	222850	87815531	673
27	May	195089	89830113	863
29	October	196830	78370242	666
29	May	176667	81482121	832
31	October	175287	69396649	649
31	May	159025	72550927	801

Table B.13: October circular scaffold reassembly statistics. Coverage denotes fold-coverage of each scaffold by metagenome sequence reads. Marine euryarchaeal virus genomes are noted (MEV1–4). See Figure A.19 for a visualization of these 59 scaffolds.

Scaffold	Contigs	Length	Coverage	GC %	N50	Note
000	118	148 Kbp	42.5×	43.0%	2292	
001	69	46 Kbp	33.5×	34.5%	990	
002	65	69 Kbp	59.5×	54.3%	1755	
003	59	51 Kbp	34.4×	44.3%	1423	
004	57	105 Kbp	49.0×	47.0%	3363	MEV1
005	54	57 Kbp	50.6×	54.0%	1485	
006	53	128 Kbp	163.2×	45.0%	4628	MEV4
007	54	30 Kbp	31.3×	35.4%	702	
008	55	47 Kbp	39.7×	40.8%	1511	
009	54	58 Kbp	38.7×	43.6%	1885	
010	52	52 Kbp	34.3×	46.4%	1399	
011	50	97 Kbp	98.8×	34.7%	3078	
012	49	28 Kbp	29.1×	41.9%	771	
013	47	34 Kbp	33.1×	54.3%	930	
014	45	66 Kbp	52.4×	47.2%	2704	
015	44	37 Kbp	34.9×	47.2%	1295	
016	44	29 Kbp	29.0×	44.1%	968	
017	43	39 Kbp	74.1×	38.1%	1374	
018	41	26 Kbp	35.0×	33.5%	989	
019	37	38 Kbp	89.7×	32.1%	1441	
020	37	27 Kbp	116.9×	39.1%	1053	
021	35	26 Kbp	43.6×	55.4%	1389	
022	36	40 Kbp	55.8×	41.7%	1547	
023	34	65 Kbp	66.5×	44.6%	3833	MEV3
024	33	53 Kbp	47.2×	43.3%	2185	
025	31	44 Kbp	122.7×	57.1%	2508	
026	32	40 Kbp	79.1×	42.7%	1874	
027	32	56 Kbp	104.6×	39.9%	4013	
028	31	58 Kbp	104.2×	35.3%	3056	
029	31	38 Kbp	57.8×	49.5%	2021	
030	29	36 Kbp	53.3×	42.5%	2124	
031	29	39 Kbp	59.7×	44.5%	2634	
032	30	41 Kbp	95.6×	43.0%	2952	
033	28	58 Kbp	62.8×	37.4%	3333	
034	29	29 Kbp	58.4×	36.2%	1785	
035	28	32 Kbp	43.9×	40.7%	1817	
036	28	30 Kbp	44.3×	50.9%	1806	
037	25	46 Kbp	207.8×	47.7%	3685	
038	25	32 Kbp	113.0×	35.1%	1667	
039	26	26 Kbp	44.2×	36.5%	1373	
040	24	35 Kbp	62.5×	41.6%	2830	
041	26	108 Kbp	87.2×	46.0%	7011	MEV2
042	24	39 Kbp	105.9×	49.2%	2985	
043	24	25 Kbp	49.7×	34.2%	2177	
044	23	32 Kbp	60.2×	37.6%	3089	
045	24	31 Kbp	51.2×	45.3%	2289	
046	23	39 Kbp	64.4×	43.6%	3379	
047	21	41 Kbp	122.0×	46.7%	5320	
048	20	57 Kbp	92.7×	44.7%	4664	

(Continued on next page)

(Continued from previous page)

Scaffold	Contigs	Length	Coverage	GC %	N50	Note
049	20	34 Kbp	87.2×	44.2%	3079	
050	19	38 Kbp	63.5×	42.3%	2659	
051	19	42 Kbp	46.3×	47.1%	3175	
052	17	47 Kbp	103.2×	47.0%	3694	
053	17	26 Kbp	41.0×	36.6%	2412	
054	14	30 Kbp	59.6×	50.9%	5284	
055	11	35 Kbp	76.4×	42.3%	7148	
056	11	38 Kbp	103.6×	43.6%	20394	
057	7	26 Kbp	110.1×	41.4%	18514	
058	6	81 Kbp	157.1×	47.9%	21460	

Table B.14: May circular scaffold reassembly statistics. Coverage denotes fold-coverage of each scaffold by metagenome sequence reads. A marine euryarchaeal virus genome is noted (MEV5). See Figure A.20 for a visualization of these 91 scaffolds.

Scaffold	Contigs	Length	Coverage	GC %	N50	Note
000	368	433 Kbp	64.7×	37.2%	1900	
001	129	137 Kbp	39.1×	38.8%	1754	
002	128	125 Kbp	34.6×	39.9%	1549	
003	124	143 Kbp	36.7×	51.1%	1652	
004	95	104 Kbp	71.0×	37.2%	1900	
005	65	72 Kbp	31.6×	44.1%	1550	
006	56	59 Kbp	50.8×	36.4%	1817	
007	52	75 Kbp	35.4×	45.7%	2008	
008	50	61 Kbp	128.2×	40.4%	2118	
009	48	78 Kbp	70.5×	38.0%	3338	
010	44	127 Kbp	62.0×	45.0%	4857	MEV5
011	45	49 Kbp	71.0×	36.4%	1467	
012	42	35 Kbp	26.6×	43.5%	1098	
013	41	84 Kbp	71.7×	40.8%	3210	
014	40	65 Kbp	71.0×	36.2%	2907	
015	40	75 Kbp	107.6×	40.7%	2881	
016	39	50 Kbp	164.7×	51.9%	2304	
017	38	33 Kbp	54.6×	43.5%	1076	
018	39	45 Kbp	43.1×	47.5%	2080	
019	37	62 Kbp	76.7×	43.4%	2953	
020	34	31 Kbp	31.5×	49.6%	1240	
021	36	31 Kbp	65.0×	43.5%	1564	
022	35	39 Kbp	93.8×	30.1%	1614	
023	33	28 Kbp	89.6×	35.6%	1450	
024	32	55 Kbp	79.8×	35.1%	3050	
025	32	45 Kbp	50.6×	56.2%	2074	
026	32	38 Kbp	254.2×	36.8%	2043	
027	33	34 Kbp	26.8×	41.4%	1561	
028	31	38 Kbp	75.7×	36.2%	2001	
029	31	32 Kbp	68.7×	44.4%	1936	
030	29	43 Kbp	95.2×	37.8%	1961	
031	30	87 Kbp	59.4×	41.6%	5347	
032	30	33 Kbp	63.4×	37.2%	2121	
033	30	68 Kbp	93.1×	35.7%	4140	
034	29	59 Kbp	169.3×	46.8%	4846	
035	29	41 Kbp	41.5×	43.5%	2308	
036	27	60 Kbp	146.8×	38.9%	4926	
037	27	36 Kbp	72.7×	35.2%	2081	
038	28	45 Kbp	67.5×	43.8%	2664	
039	28	44 Kbp	30.5×	46.2%	2658	
040	28	34 Kbp	99.0×	33.1%	1682	
041	26	28 Kbp	40.5×	40.3%	1965	
042	25	51 Kbp	37.6×	42.8%	2790	
043	25	63 Kbp	71.3×	37.8%	5372	
044	25	39 Kbp	39.4×	41.4%	1880	
045	26	45 Kbp	40.9×	45.0%	4509	
046	25	29 Kbp	33.2×	44.6%	1865	
047	23	24 Kbp	49.5×	42.8%	1994	
048	26	36 Kbp	43.5×	35.5%	2600	

(Continued on next page)

(Continued from previous page)

Scaffold	Contigs	Length	Coverage	GC %	N50	Note
049	25	39 Kbp	33.5×	43.2%	3042	
050	24	30 Kbp	76.2×	33.7%	1803	
051	24	42 Kbp	36.0×	48.8%	2316	
052	25	38 Kbp	56.6×	34.1%	2593	
053	24	100 Kbp	288.4×	41.1%	7307	
054	22	35 Kbp	148.4×	32.0%	2361	
055	22	25 Kbp	35.3×	40.3%	1677	
056	22	37 Kbp	55.6×	47.0%	2632	
057	21	24 Kbp	37.1×	55.5%	2200	
058	20	28 Kbp	31.5×	45.8%	2038	
059	22	25 Kbp	41.3×	46.7%	1488	
060	21	25 Kbp	40.0×	42.1%	1819	
061	20	150 Kbp	79.4×	42.2%	9730	
062	21	32 Kbp	36.8×	39.3%	3348	
063	19	68 Kbp	110.2×	37.9%	9077	
064	20	31 Kbp	33.3×	43.7%	2988	
065	19	38 Kbp	70.9×	42.8%	4164	
066	18	34 Kbp	145.6×	33.2%	3356	
067	19	37 Kbp	83.0×	50.8%	3044	
068	20	33 Kbp	49.7×	44.9%	3383	
069	20	50 Kbp	204.6×	38.5%	4672	
070	18	24 Kbp	57.8×	50.7%	2096	
071	19	32 Kbp	41.2×	45.3%	2611	
072	18	28 Kbp	1175.9×	30.6%	3417	
073	17	38 Kbp	82.1×	53.4%	2898	
074	17	30 Kbp	41.7×	44.6%	3775	
075	16	33 Kbp	178.7×	51.3%	4884	
076	15	43 Kbp	1133.0×	41.6%	4471	
077	15	40 Kbp	81.8×	51.1%	4158	
078	14	36 Kbp	266.4×	46.4%	7883	
079	15	26 Kbp	52.1×	48.1%	2833	
080	14	28 Kbp	70.2×	36.1%	3198	
081	15	66 Kbp	140.7×	52.2%	7667	
082	14	30 Kbp	262.9×	32.0%	4673	
083	12	36 Kbp	77.7×	45.9%	5540	
084	11	55 Kbp	80.5×	39.9%	11318	
085	11	37 Kbp	77.7×	48.2%	7404	
086	10	26 Kbp	95.7×	34.5%	4088	
087	10	45 Kbp	182.4×	36.3%	10122	
088	7	27 Kbp	177.3×	52.0%	5147	
089	8	36 Kbp	84.2×	47.1%	4217	
090	5	32 Kbp	859.0×	34.0%	13554	

Table B.15: Inter-group protein homologs. Based on pairwise reciprocal BLAST hits found with an e-value cutoff of 10^{-5}. MG2 is the marine group II *Euryarchaeote* genome (`CM001443`). Table entries are locus_id values, so in the "MG2" column the value "`0752`" corresponds to the gene with locus_id of "`MG2_0752`". See Table 3.2 for a condensed version.

	Group 1			Group 2		Group 3			Host
Product	MEV1	MEV2	GG3SC144	MEV3	GG2SC877	MEV4	MEV5	GG2SC16	MG2
RBH Shared among all virus groups									
thermosome chaperone or *GroEL*	0064	0057	0053	*0046*	*0050*	0019	0019	0024	0752
Virus Groups 1 and 2 RBH									
terminase, large subunit	0092	0085	0081	0002	0001	–	–	–	–
phage / plasmid-like protein	–	0024	0001	0022	–	–	–	–	–
hypothetical protein	–	0029	–	0026	–	–	–	–	–
DNA methylase	0047	–	–	0027	0048	–	–	–	–
replication factor C small subunit	0051	0043	0037	0044	0036	–	–	–	1397
DNA polymerase elongation subunit	0056	0048	0042	0053	0046	–	–	–	1733
putative exodeoxyribonuclease	0058	0050	0046	0056	0053	–	–	–	–
hypothetical protein	0059	0052	0049	0058	0054	–	–	–	–
putative DNA repair nuclease	0061	0054	0050	0059	0057	–	–	–	–
putative phage tail fiber protein	0067	0059	0057	0074	0069	–	–	–	–
Virus Groups 1 and 3 RBH									
hypothetical protein	0100	0093	0089	–	–	0055	0053	0055	–
putative P-loop NTPase	0102	0094	0091	–	–	0056	0054	0056	–
hypothetical protein	0104	0096	0092	–	–	0059	0057	0059	–
hypothetical protein	–	0019	0017	–	–	0126	0117	–	–
hypothetical protein	–	–	0026	–	–	0162	0157	–	–
hypothetical protein	–	–	0057	–	–	0181	0177	–	–
phage / plasmid-like protein	0031	0039	–	–	–	–	–	0123	–
hypothetical protein	0044	–	–	–	–	–	–	0112	–
Virus Groups 2 and 3 RBH									
DNA polymerase sliding clamp subunit	–	–	–	0023	–	0144	0138	0133	1519
DNA methylase	–	–	–	0051	0047	0178	0174	0135	0345
Non-shared MG-II RBH orthologs									
hypothetical protein	–	0023	–	–	–	–	–	–	0900
hypothetical protein	–	–	–	–	0067	–	–	–	0911
hypothetical protein	–	–	–	–	0072	–	–	–	1689
pentapeptide repeats family protein	–	–	–	–	–	0071	0069	–	0480
DNA-directed RNA polymerase, subunit A	–	–	–	–	–	0175	0171	–	0884
winged helix-turn-helix DNA-bind protein	–	–	–	–	–	0116	0107	–	1006
putative hemolysin protein or related	–	–	–	–	–	0127	0118	–	1126
hypothetical protein	–	–	–	–	–	–	–	0101	0973
winged helix-turn-helix DNA-bind protein	–	–	–	–	–	–	–	0122	1581

Table B.16: Protein homologs and predicted products for Group 1 virus sequences: MEV1, MEV2 and GG3SC144. Homology based on pairwise reciprocal BLAST hits found with an e-value cutoff of 10^{-5}. Numbers in first 3 columns correspond to the suffix of the locus_id of the predicted gene coding for the homolog (e.g. `MEV1_0001`). See methods for annotation details.

MEV1	MEV2	GG3SC144	Product
0001			hypothetical protein
0002			hypothetical protein
0003			hypothetical protein
0004			hypothetical protein
0005			hypothetical protein
0006	0005		hypothetical protein
0007			hypothetical protein
0008			hypothetical protein
0009	0013		hypothetical protein
0010	0026		hypothetical protein
0011			hypothetical protein
0012	0009		hypothetical protein
0013	0023		hypothetical protein
0014			hypothetical protein
0015			hypothetical protein
0016			hypothetical protein
0017			hypothetical protein
0018			hypothetical protein
0019			hypothetical protein
0020			hypothetical protein
0021			hypothetical protein
0022	0012	0011	hypothetical protein
0023			hypothetical protein
0024			hypothetical protein
0025	0034		**putative DNA helicase**
0026			hypothetical protein
0027			hypothetical protein
0028			hypothetical protein
0029	0021	0017	hypothetical protein
0030			hypothetical protein
0031	0039		**phage/plasmid-like protein**
0032			hypothetical protein
0033	0040	0040	hypothetical protein
0034			hypothetical protein
0035			hypothetical protein
0036			hypothetical protein
0037			hypothetical protein
0038			hypothetical protein
0039			hypothetical protein
0040			hypothetical protein
0041			hypothetical protein
0042	0041		hypothetical protein
0043			hypothetical protein
0044			hypothetical protein
0045			hypothetical protein
0046			hypothetical protein
0047			**DNA methylase**
0048			hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV1	MEV2	GG3SC144	Product
0049			hypothetical protein
0050	0042		hypothetical protein
0051	0043	0037	**replication factor C small subunit**
0052	0044	0038	hypothetical protein
0053	0045	0041	hypothetical protein
0054	0046	0043	hypothetical protein
0055			hypothetical protein
0056	0048	0042	**DNA polymerase elongation subunit (family B)**
0057			hypothetical protein
0058	0050	0046	**putative exodeoxyribonuclease**
0059	0052	0049	hypothetical protein
0060			**putative winged helix-turn-helix DNA-binding protein**
0061	0054	0050	**putative DNA repair nuclease**
0062	0055	0051	hypothetical protein
0063	0056	0052	**ribonuclease H-like domain protein**
0064	0057	0053	**thermosome chaperone (cpn60)**
0065			hypothetical protein
0066			hypothetical protein
0067	0059		**putative phage tail fiber protein**
0068	0061	0059	hypothetical protein
0069	0062	0060	hypothetical protein
0070	0063		hypothetical protein
0071	0064	0061	hypothetical protein
0072	0065		hypothetical protein
0073	0067	0064	hypothetical protein
0074			hypothetical protein
0075	0068	0066	hypothetical protein
0076	0069	0067	hypothetical protein
0077	0070	0069	hypothetical protein
0078			hypothetical protein
0079	0072		hypothetical protein
0080			hypothetical protein
0081	0075	0071	hypothetical protein
0082	0076	0072	hypothetical protein
0083	0077	0073	**major capsid protein**
0084	0078	0074	**prohead protease**
0085	0079	0075	hypothetical protein
0086	0080	0077	hypothetical protein
0087	0081		hypothetical protein
0088	0082	0078	hypothetical protein
0089	0083	0079	**phage portal protein**
0090	0084	0080	**putative homing endonuclease**
0091			hypothetical protein
0092	0085	0081	**terminase DNA packaging enzyme large subunit**
0093			hypothetical protein
0094	0088	0083	hypothetical protein
0095	0089	0084	hypothetical protein
0096	0090	0086	**putative RNA polymerase sigma factor protein**
0097			hypothetical protein
0098	0091	0087	hypothetical protein
0099	0092	0088	hypothetical protein
0100	0093	0089	hypothetical protein
0101			hypothetical protein
0102	0094	0091	**hypothetical P-loop NTPase**

(Continued on next page)

(Continued from previous page)

MEV1	MEV2	GG3SC144	Product
0103	0095	0092	hypothetical protein
0104	0096	0093	hypothetical protein
0105	0097		hypothetical protein
0106			hypothetical protein
0107	0099	0096	hypothetical protein
0108	0100		hypothetical protein
0109			hypothetical protein
0110	0102	0095	hypothetical protein
0111	0105		**putative pyruvate carboxyltransferase**
0112			hypothetical protein
0113			hypothetical protein
0114			hypothetical protein
0115			hypothetical protein
0116			hypothetical protein
0117			hypothetical protein
0118			hypothetical protein
0119			hypothetical protein
	0001		hypothetical protein
	0002		hypothetical protein
	0003		hypothetical protein
	0004		hypothetical protein
	0006		hypothetical protein
	0007		hypothetical protein
	0008		hypothetical protein
	0010		hypothetical protein
	0011		hypothetical protein
	0014		hypothetical protein
	0015		hypothetical protein
	0016		hypothetical protein
	0017		hypothetical protein
	0018		hypothetical protein
	0019		hypothetical protein
	0020		hypothetical protein
	0022		hypothetical protein
	0024	0001	**phage/plasmid-like protein**
	0025		hypothetical protein
	0027		hypothetical protein
	0028		hypothetical protein
	0029		hypothetical protein
	0030		hypothetical protein
	0031		hypothetical protein
	0032		hypothetical protein
	0033		hypothetical protein
	0035		hypothetical protein
	0036		hypothetical protein
	0037		hypothetical protein
	0038		hypothetical protein
	0047		hypothetical protein
	0049	0044	hypothetical protein
	0051		hypothetical protein
	0053		hypothetical protein
	0058		hypothetical protein
	0060		hypothetical protein
	0066		hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV1	MEV2	GG3SC144	Product
	0071		hypothetical protein
	0073		hypothetical protein
	0074		hypothetical protein
	0086		hypothetical protein
	0087		hypothetical protein
	0098		hypothetical protein
	0101		hypothetical protein
	0103		hypothetical protein
	0104		hypothetical protein
	0106		hypothetical protein
	0107		hypothetical protein
	0108		hypothetical protein
	0109		hypothetical protein
		0002	hypothetical protein
		0003	hypothetical protein
		0004	hypothetical protein
		0005	hypothetical protein
		0006	hypothetical protein
		0007	hypothetical protein
		0008	hypothetical protein
		0009	hypothetical protein
		0010	hypothetical protein
		0012	hypothetical protein
		0013	hypothetical protein
		0014	hypothetical protein
		0015	hypothetical protein
		0016	hypothetical protein
		0018	hypothetical protein
		0019	hypothetical protein
		0020	hypothetical protein
		0021	hypothetical protein
		0022	hypothetical protein
		0023	hypothetical protein
		0024	hypothetical protein
		0025	hypothetical protein
		0026	hypothetical protein
		0027	hypothetical protein
		0028	hypothetical protein
		0029	hypothetical protein
		0030	hypothetical protein
		0031	hypothetical protein
		0032	hypothetical protein
		0033	hypothetical protein
		0034	hypothetical protein
		0035	hypothetical protein
		0036	hypothetical protein
		0039	hypothetical protein
		0045	hypothetical protein
		0047	hypothetical protein
		0048	hypothetical protein
		0054	hypothetical protein
		0055	hypothetical protein
		0056	hypothetical protein
		0057	hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV1	MEV2	GG3SC144	Product
		0058	hypothetical protein
		0062	hypothetical protein
		0063	hypothetical protein
		0065	hypothetical protein
		0068	hypothetical protein
		0070	hypothetical protein
		0076	hypothetical protein
		0082	hypothetical protein
		0085	hypothetical protein
		0090	hypothetical protein
		0094	hypothetical protein
		0097	hypothetical protein
		0098	hypothetical protein
		0099	hypothetical protein
		0100	hypothetical protein
		0101	hypothetical protein
		0102	hypothetical protein
		0103	hypothetical protein
		0104	hypothetical protein

Table B.17: Protein homologs and predicted products for Group 2 virus sequences: MEV3 and GG2SC877. Homology based on pairwise reciprocal BLAST hits found with an e-value cutoff of 10^{-5}. Numbers in first 2 columns correspond to the suffix of the locus_id of the predicted gene coding for the homolog (e.g. `MEV3_0001`). See methods for annotation details.

MEV3	GG2SC877	Product
0001		**metallo-dependent phosphatase**
0002	0001	**terminase large subunit**
0003		hypothetical protein
0004		hypothetical protein
0005		hypothetical protein
0006		hypothetical protein
0007		hypothetical protein
0008		hypothetical protein
0009		hypothetical protein
0010		hypothetical protein
0011		hypothetical protein
0012		hypothetical protein
0013		hypothetical protein
0014		hypothetical protein
0015		hypothetical protein
0016		hypothetical protein
0017		hypothetical protein
0018		**antirestriction protein (ArdA-like)**
0019		hypothetical protein
0020		hypothetical protein
0021		hypothetical protein
0022		**phage/plasmid-like protein**
0023		**DNA polymerase sliding clamp subunit**
0024		hypothetical protein
0025		hypothetical protein
0026		hypothetical protein
0027	0048	**DNA methylase**
0028		hypothetical protein
0029		hypothetical protein
0030	0026	hypothetical protein
0031		hypothetical protein
0032		hypothetical protein
0033	0027	**putative P-loop NTPase**
0034	0028	hypothetical protein
0035	0029	hypothetical protein
0036	0031	hypothetical protein
0037		hypothetical protein
0038	0033	hypothetical protein
0039		hypothetical protein
0040		hypothetical protein
0041	0019	hypothetical protein
0042		hypothetical protein
0043		hypothetical protein
0044	0036	**replication factor C small subunit**
0045	0039	hypothetical protein
0046	0050	**GroEL-like chaperone (cpn60)**
0047	0040	hypothetical protein
0048		hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV3	GG2SC877	Product
0049	0043	hypothetical protein
0050		hypothetical protein
0051	0047	**DNA methylase**
0052	0045	hypothetical protein
0053	0046	**DNA polymerase elongation subunit (family B)**
0054		hypothetical protein
0055		hypothetical protein
0056	0053	**putative exodeoxyribonuclease**
0057		hypothetical protein
0058	0054	hypothetical protein
0059	0057	**putative DNA repair nuclease**
0060	0058	hypothetical protein
0061		hypothetical protein
0062	0060	**phage tail tape measure protein**
0063		hypothetical protein
0064	0061	hypothetical protein
0065	0062	hypothetical protein
0066	0064	hypothetical protein
0067	0065	hypothetical protein
0068		hypothetical protein
0069	0067	hypothetical protein
0070		hypothetical protein
0071		**SAP-domain protein**
0072		hypothetical protein
0073		hypothetical protein
0074	0069	**putative phage tail fiber protein**
0075	0070	hypothetical protein
0076	0071	hypothetical protein
0077		hypothetical protein
0078		hypothetical protein
0079	0073	hypothetical protein
0080	0074	hypothetical protein
0081	0075	**putative phage protein**
0082		**putative Acyl-CoA N-acyltransferase**
0083	0078	hypothetical protein
0084	0079	**phage portal protein**
	0002	hypothetical protein
	0003	hypothetical protein
	0004	hypothetical protein
	0005	hypothetical protein
	0006	hypothetical protein
	0007	hypothetical protein
	0008	hypothetical protein
	0009	hypothetical protein
	0010	hypothetical protein
	0011	hypothetical protein
	0012	hypothetical protein
	0013	hypothetical protein
	0014	hypothetical protein
	0015	hypothetical protein
	0016	hypothetical protein
	0017	hypothetical protein
	0018	hypothetical protein
	0020	hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV3	GG2SC877	Product
	0021	hypothetical protein
	0022	hypothetical protein
	0023	hypothetical protein
	0024	hypothetical protein
	0025	hypothetical protein
	0030	hypothetical protein
	0032	hypothetical protein
	0034	hypothetical protein
	0035	hypothetical protein
	0037	hypothetical protein
	0038	hypothetical protein
	0041	hypothetical protein
	0042	hypothetical protein
	0044	hypothetical protein
	0049	hypothetical protein
	0051	hypothetical protein
	0052	hypothetical protein
	0055	hypothetical protein
	0056	hypothetical protein
	0059	**phage tail tape measure protein**
	0063	hypothetical protein
	0066	hypothetical protein
	0068	hypothetical protein
	0072	hypothetical protein
	0076	**putative phage protein**
	0077	hypothetical protein
	0080	hypothetical protein
	0081	hypothetical protein
	0082	hypothetical protein

Table B.18: Protein homologs and predicted products for Group 3 virus sequences: ME4, MEV5 and GG2SC16. Homology based on pairwise reciprocal BLAST hits found with an e-value cutoff of 10^{-5}. Numbers in first 3 columns correspond to the suffix of the locus_id of the predicted gene coding for the homolog (e.g. `MEV4_0001`). See methods for annotation details.

MEV4	MEV5	GG2SC16	Product
0001	0001	0001	**putative winged helix-turn-helix DNA-binding protein**
0002	0002	0002	hypothetical protein
0003	0003		hypothetical protein
0004	0004		hypothetical protein
0005	0005	0005	hypothetical protein
0006	0006	0007	hypothetical protein
0007	0007	0008	hypothetical protein
0008	0008	0030	hypothetical protein
0009	0009	0012	hypothetical protein
0010	0010	0013	hypothetical protein
0011	0011	0014	hypothetical protein
0012	0012	0016	**metallo-dependent phosphatase**
0013	0013		hypothetical protein
0014	0014	0017	hypothetical protein
0015	0015		hypothetical protein
0016	0016	0019	hypothetical protein
0017	0017	0020	hypothetical protein
0018	0018	0023	hypothetical protein
0019	0019	0024	**thermosome chaperone (cpn60)**
0020	0020		**DNA methylase**
0021	0021	0026	**putative P-loop NTPase**
0022	0022		**putative ribbon-helix-helix protein**
0023	0023		hypothetical protein
0024	0024	0028	hypothetical protein
0025	0025		hypothetical protein
0026	0026		hypothetical protein
0027	0027		hypothetical protein
0028	0028		hypothetical protein
0029	0029	0033	hypothetical protein
0030	0030		hypothetical protein
0031	0031		hypothetical protein
0032	0032		hypothetical protein
0033	0033	0036	hypothetical protein
0034	0034	0037	hypothetical protein
0035	0035	0038	hypothetical protein
0036	0036		hypothetical protein
0037	0037	0041	hypothetical protein
0038	0038		hypothetical protein
0039			hypothetical protein
0040	0040		hypothetical protein
0041			hypothetical protein
0042	0041		hypothetical protein
0043	0042	0044	**phage integrase protein**
0044	0043		hypothetical protein
0045	0044		hypothetical protein
0046	0045		hypothetical protein
0047	0046		hypothetical protein
0048	0047	0048	hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV4	MEV5	GG2SC16	Product
0049	0048	0050	hypothetical protein
0050			hypothetical protein
0051	0049		hypothetical protein
0052	0050		hypothetical protein
0053	0051	0054	hypothetical protein
0054	0052		hypothetical protein
0055	0053	0055	hypothetical protein
0056	0054	0056	**putative P-loop NTPase**
0057	0055		hypothetical protein
0058	0056	0058	hypothetical protein
0059	0057	0059	hypothetical protein
0060	0058		hypothetical protein
0061	0059		hypothetical protein
0062	0060	0062	hypothetical protein
0063	0061	0063	**glycosyltransferase**
0064	0062	0064	**glycosyl transferase group 1**
0065	0063		hypothetical protein
0066	0064		hypothetical protein
0067	0065		hypothetical protein
0068	0066		**putative transcription initiation factor TFIIB**
0069	0067		hypothetical protein
0070	0068		hypothetical protein
0071	0069		**pentapeptide repeats family protein**
0072	0071		hypothetical protein
0073			hypothetical protein
0074	0070		hypothetical protein
0075			hypothetical protein
0076	0073		hypothetical protein
0077	0072		hypothetical protein
0078			hypothetical protein
0079			hypothetical protein
0080	0074		hypothetical protein
0081	0075		hypothetical protein
0082	0076		hypothetical protein
0083			hypothetical protein
0084	0077		hypothetical protein
0085			hypothetical protein
0086	0078		hypothetical protein
0087	0079		hypothetical protein
0088	0080		hypothetical protein
0089	0081		hypothetical protein
0090	0082		hypothetical protein
0091	0083		hypothetical protein
0092	0084		hypothetical protein
0093	0085	0102	hypothetical protein
0094	0086		hypothetical protein
0095	0087		hypothetical protein
0096	0088		hypothetical protein
0097			hypothetical protein
0098	0089		hypothetical protein
0099	0090		hypothetical protein
0100	0091		hypothetical protein
0101	0092		**putative P-loop NTPase**
0102	0094		hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV4	MEV5	GG2SC16	Product
0103	0095		hypothetical protein
0104	0096		hypothetical protein
0105	0097		hypothetical protein
0106	0098		hypothetical protein
0107	0099		hypothetical protein
0108	0100		hypothetical protein
0109	0101	0094	**DNA methylase**
0110			hypothetical protein
0111	0102		hypothetical protein
0112	0103		hypothetical protein
0113	0104		hypothetical protein
0114	0105		hypothetical protein
0115	0106		hypothetical protein
0116	0107	0122	**putative winged helix-turn-helix transcription repressor**
0117	0108		hypothetical protein
0118	0109		hypothetical protein
0119	0110		hypothetical protein
0120	0111		hypothetical protein
0121	0113	0130	hypothetical protein
0122	0114		hypothetical protein
0123	0115		hypothetical protein
0124			hypothetical protein
0125	0116		hypothetical protein
0126	0117		hypothetical protein
0127	0118		**putative hemolysin or related protein containing CBS domains**
0128	0119		hypothetical protein
0129	0120		hypothetical protein
0130	0121		hypothetical protein
0131	0122		hypothetical protein
0132	0123	0103	**DNA methylase**
0133	0126		hypothetical protein
0134	0128		hypothetical protein
0135	0129		hypothetical protein
0136	0130	0090	hypothetical protein
0137	0131		hypothetical protein
0138	0132		hypothetical protein
0139	0133	0114	hypothetical protein
0140	0134		hypothetical protein
0141	0135		hypothetical protein
0142	0136		hypothetical protein
0143	0137		hypothetical protein
0144	0138		**DNA polymerase sliding clamp subunit**
0145	0139		hypothetical protein
0146	0140		hypothetical protein
0147	0141		hypothetical protein
0148	0142		hypothetical protein
0149	0143	0118	hypothetical protein
0150	0144	0135	**DNA methylase**
0151	0145		hypothetical protein
0152	0146		hypothetical protein
0153	0147		hypothetical protein
0154	0148		hypothetical protein
0155	0149		hypothetical protein
0156	0150		hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV4	MEV5	GG2SC16	Product
0157	0152		hypothetical protein
0158	0153		hypothetical protein
0159	0154		hypothetical protein
0160	0155		hypothetical protein
0161	0156		hypothetical protein
0162	0157		hypothetical protein
0163	0158		hypothetical protein
0164	0159		hypothetical protein
0165	0160		hypothetical protein
0166	0161		hypothetical protein
0167	0162		hypothetical protein
0168	0163		hypothetical protein
0169	0164		hypothetical protein
0170	0165		hypothetical protein
0171	0166		hypothetical protein
0172	0167		hypothetical protein
0173	0168		hypothetical protein
0174	0170		hypothetical protein
0175	0171	0139	**DNA-directed RNA polymerase**
0176	0172	0141	hypothetical protein
0177	0173	0142	**Ribonuclease H**
0178	0174		**DNA methylase**
0179	0175	0144	hypothetical protein
0180	0176		hypothetical protein
0181	0177		hypothetical protein
0182	0179	0147	hypothetical protein
0183	0180	0145	hypothetical protein
0184	0181	0148	hypothetical protein
0185	0183		hypothetical protein
0186	0184	0149	hypothetical protein
0187	0185		hypothetical protein
0188	0186		hypothetical protein
0189			hypothetical protein
	0039		hypothetical protein
	0093		hypothetical protein
	0112		hypothetical protein
	0124		hypothetical protein
	0125		hypothetical protein
	0127		hypothetical protein
	0151		hypothetical protein
	0169		hypothetical protein
	0178		hypothetical protein
	0182		hypothetical protein
	0187		hypothetical protein
		0003	hypothetical protein
		0004	hypothetical protein
		0006	hypothetical protein
		0009	hypothetical protein
		0010	hypothetical protein
		0011	hypothetical protein
		0015	hypothetical protein
		0018	hypothetical protein
		0021	hypothetical protein
		0022	hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV4	MEV5	GG2SC16	Product
		0025	**thermosome chaperone (cpn60)**
		0027	hypothetical protein
		0029	hypothetical protein
		0031	hypothetical protein
		0032	hypothetical protein
		0034	hypothetical protein
		0035	hypothetical protein
		0039	hypothetical protein
		0040	hypothetical protein
		0042	hypothetical protein
		0043	hypothetical protein
		0045	hypothetical protein
		0046	hypothetical protein
		0047	hypothetical protein
		0049	hypothetical protein
		0051	hypothetical protein
		0052	hypothetical protein
		0053	hypothetical protein
		0057	hypothetical protein
		0060	hypothetical protein
		0061	hypothetical protein
		0065	hypothetical protein
		0066	hypothetical protein
		0067	hypothetical protein
		0068	hypothetical protein
		0069	hypothetical protein
		0070	hypothetical protein
		0071	hypothetical protein
		0072	hypothetical protein
		0073	hypothetical protein
		0074	hypothetical protein
		0075	hypothetical protein
		0076	hypothetical protein
		0077	hypothetical protein
		0078	hypothetical protein
		0079	hypothetical protein
		0080	hypothetical protein
		0081	hypothetical protein
		0082	hypothetical protein
		0083	hypothetical protein
		0084	hypothetical protein
		0085	hypothetical protein
		0086	hypothetical protein
		0087	hypothetical protein
		0088	hypothetical protein
		0089	hypothetical protein
		0091	hypothetical protein
		0092	hypothetical protein
		0093	hypothetical protein
		0095	hypothetical protein
		0096	hypothetical protein
		0097	hypothetical protein
		0098	hypothetical protein
		0099	hypothetical protein

(Continued on next page)

(Continued from previous page)

MEV4	MEV5	GG2SC16	Product
		0100	hypothetical protein
		0101	hypothetical protein
		0104	hypothetical protein
		0105	hypothetical protein
		0106	hypothetical protein
		0107	hypothetical protein
		0108	hypothetical protein
		0109	hypothetical protein
		0110	hypothetical protein
		0111	hypothetical protein
		0112	hypothetical protein
		0113	hypothetical protein
		0115	hypothetical protein
		0116	hypothetical protein
		0117	hypothetical protein
		0119	hypothetical protein
		0120	hypothetical protein
		0121	hypothetical protein
		0123	hypothetical protein
		0124	hypothetical protein
		0125	hypothetical protein
		0126	hypothetical protein
		0127	hypothetical protein
		0128	hypothetical protein
		0129	hypothetical protein
		0131	hypothetical protein
		0132	hypothetical protein
		0133	**DNA polymerase sliding clamp subunit**
		0134	hypothetical protein
		0136	hypothetical protein
		0137	hypothetical protein
		0138	hypothetical protein
		0140	hypothetical protein
		0143	hypothetical protein
		0146	hypothetical protein
		0150	hypothetical protein
		0151	hypothetical protein
		0152	hypothetical protein

Table B.19: Marine group II *Euryarchaeota* scaffolds assembled from the October and May metagenomes that contain genes coding for thermosome proteins. Scaffolds directly corresponding to the published MG-II genome (CM001443) are excluded. Subunits reference the marine group II thermosome paralogs noted in Figure A.29. Coverage denotes fold-coverage of the scaffold by metagenome sequence reads. Size is the genome/scaffold length in thousands of nucleotides. Homologs were determined as genes with pairwise reciprocal BLAST hits to the MG-II genome with an e-value cutoff of 10^{-5}. For graphical comparison of these scaffolds to the MG-II genome see Figures A.27 and A.28.

Name	Sample	Locus	Subunit	Coverage	GC %	Size	N50	Genes	Homologs
GG2SC223	October	0030	α	29.8×	49.3%	39	775	48	37 (77%)
GG2SC77	October	0034	β	26.9×	47.6%	56	838	58	42 (72%)
GG2SC40	October	0079	β	46.9×	48.4%	117	1487	129	57 (44%)
GG2SC62	October	0047	β	25.9×	41.8%	59	832	59	45 (76%)
GG2SC0	October	0160	γ	33.4×	48.8%	232	1172	190	163 (90%)
GG2SC551	October	0009	γ	25.1×	41.2%	20	778	20	17 (75%)
GG3SC224	May	0007	γ	26.5×	45.4%	37	1084	41	36 (88%)

Appendix C

THE SEASTAR TOOLS

C.1 Introduction

The SEAStAR tools[1] are a collection of programs which implement many data and processing intensive steps in next-generation sequencing analysis pipelines. The tools quantitatively analyze alignments to reference sequence databases (including *de novo* assembled contigs), to produce a wide variety of statistics and relationships, including producing binned, assembled genomic scaffolds. At one time, SEAStAR was an acronym for **S**elect and **E**stimate **A**bundance from **S**hort **A**ligned **R**eads. However, as described above, the tools and their capabilities have expanded well beyond that description, and so now SEAStAR is just our name for the in-house developed components of our analysis pipeline.

Our philosophy has been to create stable, modular, high-performance tools that fill important gaps we encountered between other established pre-existing tools in our pipeline. As such, you will see references to other packages throughout this document. We recognize that considerable innovation is ongoing in analysis software for next generation sequence, and so we have maintained a modular, pipelined approach to permit experimentation with, and substitution of, new tools as they become available. Due to the wide variety of different computational environments that undoubtedly exist in different labs and institutes, our focus will be on releasing stable, documented, portable and high performance *components* from our analysis pipeline, but not the pipeline scripts themselves. This is because these scripts

[1] This appendix is adapted from version 0.5.0 of the SEAStAR tools, hosted on github, and co-developed with Chris Berthiaume

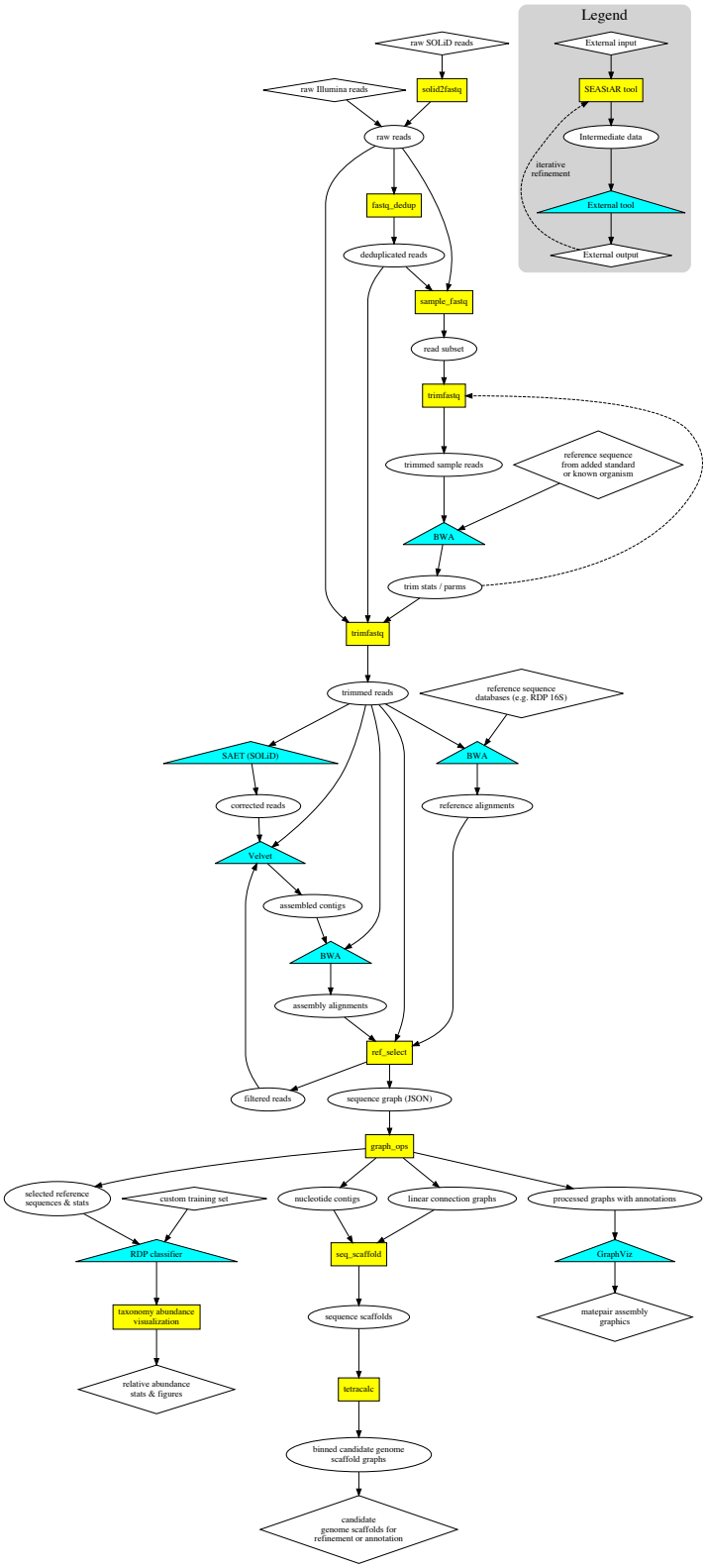

Figure C.1: SEAStAR Analysis Pipeline

will need to be highly customized for both the goals and computational environment of any specific project.

Initially our goal was to release tools to enable others to work directly with datasets generated by the Life Technologies SOLiD™; sequencing platform. We recognized that most of the components we have developed are also valuable for use with Illumina®; sequence data, or with a mixture of sequence data types, and so in this release we have generalized most components to support both colorspace and nucleotide reads.

C.1.1 Technical guidance

The next-generation sequence datasets that the SEAStAR tools are designed to work with are typically very large (>> one billion reads). All compiled components of the SEAStAR tools are multithreaded and optimized for modern multi-core processors. Many SEAStAR tools will take advantage of 8 or more cores, particularly when compressed (gzip format) input and output files are being processed. When running on such a system, you will find that these components may run considerably more than 10× faster than equivalent tools written in a scripting language such as Python. For this reason, the operation of such tools will often be "I/O bound"; that is: the performance bottleneck is the speed of the disk(s) and/or network connections connected to the workstation, and not the availability of CPU cycles. The use of compressed files and local, separate independent disks for input and output files can often relieve some of this bottleneck and further improve run times.

Several of the SEAStAR tools (as well as many other tools used by our pipeline) have very high memory requirements when working with typical next-generation sequence datasets. We note these cases in the discussion of each individual SEAStAR tool, and attempt to provide guidance about the impact different parameters will have on memory use. However, in general, doing this kind of work will require a workstation with at least 32 gigabytes of

RAM, and depending on the dataset and the types of analysis your project requires, you may require some multiple of that amount (64, 128, 256, or even 512 gigabytes) to successfully implement an analysis pipeline that meets your needs. When running these tools, it is highly advisable to set a session limit on the memory a process can attempt to allocate, to prevent machines from crashing and impacting other users who may be logged in. For example, adding the Unix shell command:

```
ulimit -v 64000000
```

to any script that invokes high memory tools will effectively prevent a 64GB machine from being unintentionally brought down (i.e. thrashing the swap file) by an application that attempts to allocate more memory than is physically available.

C.1.2 License and citation

The SEAStAR tools are open source and are currently publicly distributed under the GPL version 3 license, a copy of which is provided in a file named `COPYING` in the project root directory. By using this software, you are agreeing to be bound by the terms of this license. Please contact the authors if you are interested in discussing an alternative licensing arrangement.

If you use these tools to analyze data for publication we ask that you cite the following paper (which was the first to use and describe these tools):

> Iverson, V., Morris, R. M., Frazar, C. D., Berthiaume, C. T., Morales, R. L . and Armbrust, E. V., Untangling Genomes from Metagenomes: Revealing an Uncultured Class of Marine Euryarchaeota, *Science* 335 pp.587-590 (2012)

C.2 Quick start

This section is a quick introduction to using SEAStAR tools for some common use cases. The SEAStAR tools are "command-line" tools that run on UNIX-like operating systems. If

you are not familiar with how to use the UNIX command-line shell, you should work your way through a tutorial before proceeding. Here are a few possibilities:

- UNIX Tutorial for Beginners
- Learning the Linux Shell
- Introduction to the OS X Unix Command Line

This tutorial assumes that you have already installed the SEAStAR software. If you are working from the SEAStAR Quick Start virtual machine appliance file in VirtualBox, your environment is already set-up and ready to go, so you can skip to the next section: Read Preparation.

For SEAStAR installation instructions see the `README` file in the SEAStAR root directory. Although the SEAStAR tools can be quite memory and resource intensive, with the data used in this tutorial you should be able to comforably complete all of the steps below on a modern laptop with a dual-core processor and 2GB of RAM.

The data used in these examples are actual raw SOLiD sequence reads corresponding to the Lambda phage genome, and can be found in the `test_data` subdirectory of the SEAStAR source directory.

Create a new working directory and copy the following files from the `test_data` directory into that directory.

```
# Substitute the actual path to your SEAStAR "source directory" for
# [SEASTAR_src] in the copy commands below:

mkdir Quick_start
cd Quick_start

cp [SEASTAR_src]/test_data/lambda_reads_F3.csfasta.gz .
cp [SEASTAR_src]/test_data/lambda_reads_F3_QV.qual.gz .
cp [SEASTAR_src]/test_data/lambda_reads_R3.csfasta.gz .
cp [SEASTAR_src]/test_data/lambda_reads_R3_QV.qual.gz .
```

And make sure that your PATH environment variable points to the executable files in your SEAStAR binaries directory:

```
# Where [SEASTAR_dest] below is the fully qualified path to your
# SEAStAR build destination directory.
export PATH=$PATH:[SEASTAR_dest]/bin
```

If the PATH above is working, you should be able to run `ref_select --version` and get back something like: `ref_select version: v0.4.x`. If this doesn't work, you will need to figure out where SEAStAR is installed before proceeding so that the instructions that follow in this tutorial can find the SEAStAR tools on your computer.

NOTE: To work your way through all of the examples in this tutorial, you will need the following external tools (in addition to those that were required to build SEAStAR):

- Velvet – You need the **colorspace** version. Build it with: `make color`, see the Velvet README file for instructions.
- BWA – **Version 0.5.10**. *Important!! Because newer versions do not support the SOLiD colorspace reads used by this tutorial*
- GraphViz

Please ensure that the directories where these tools reside are also in your PATH environment variable before proceeding. You can test this by typing "`velveth_de`", "`bwa`", and "`dot -V`" at your command line. Each of these commands should write its version string (and perhaps other help messages) in response to being run.

C.2.1 Read preparation

Convert SOLiD `.csfasta` and `.qual` files to gzipped `.fastq` files. For real projects, users with no colorspace data will omit this step in their analysis pipeline, although the FASTQ

file naming conventions still need to be followed for nucleotide (e.g. Illumina) reads; see the FASTQ section for details.

```
solid2fastq -z lambda_reads lambda_conv
```

Remove presumed PCR duplicate reads, identified by mate-pairs seen more than once.

```
fastq_nodup -z -l 13 -d 2 -e 3 lambda_conv lambda_dedup
```

Trim and filter the de-duplicated FASTQ reads based on read quality, information content, and length.

```
trimfastq -z --mates_file -p 0.9 -l 34 -m 34 --add_len -e 3.0 \
   lambda_dedup lambda_trim
```

Randomly sample reads from `.fastq` files, retaining approximately 10% of the original reads. Randomly subsampled read sets are useful for quickly tuning the parameters of `trimfastq` for a given set of reads against some reference. We won't do that here, but tuning the `trimfastq -p` setting is important for getting clean assemblies and maximizing the useful information in your sequence data.

```
samplefastq -z -f 0.1 lambda_trim lambda_samp
```

C.2.2 De novo contig assembly

If you have installed the colorspace aware build of the *de novo* assembly tool Velvet, it can be used to assemble lambda-phage colorspace contigs:

```
velveth_de lambda_asm/ 19 -fastq.gz -shortPaired \
   lambda_trim.mates.fastq.gz -short lambda_trim.single.fastq.gz > \
   lambda_asm.velveth_de.log 2>&1

velvetg_de lambda_asm/ -scaffolding no -read_trkg no -ins_length \
   auto -ins_length_sd auto -exp_cov 50 -cov_cutoff 5 \
   -min_contig_lgth 50 > lambda_asm.velvetg_de.log 2>&1
```

There will now be a subdirectory called `lambda_asm` with a file called `contigs.fa` in it. These are the colorspace contigs we will use in the next section.

C.2.3 Aligning reads to a reference

For this example, you need the short read aligner BWA (version <= 0.5.10). It can be used to align the de-duplicated and trimmed lambda FASTQ reads against the lambda-phage colorspace contigs. IMPORTANT: This quickstart example will not work with versions of BWA version 0.6.0 or newer because colorspace support was removed from BWA at that point. For your own work, if you are using nucleotide reads, you are free to use any alignment software you wish that produces standard SAM files.

```
# Because these are colorspace contigs and BWA expects nucleotide
# reference sequences, we need to convert the colorspace contigs to
# nucleotides using a single starting nucleotide (this will be wrong
# 3/4 of the time, but it doesn't matter because BWA immediately
# converts the reference back to colorspace internally...)

csfasta2ntfasta.awk lambda_asm/contigs.fa > lambda_contigs.fna
bwa index -a is -c lambda_contigs.fna
bwa aln -c -n 0.001 -l 18 lambda_contigs.fna \
   lambda_trim.read1.fastq.gz > lambda_trim.read1.sai
bwa samse -n 1000000 lambda_contigs.fna lambda_trim.read1.sai \
   lambda_trim.read1.fastq.gz 2>lambda_trim.read1.samse.log > \
   lambda_trim.read1.sam
bwa aln -c -n 0.001 -l 18 lambda_contigs.fna \
   lambda_trim.read2.fastq.gz > lambda_trim.read2.sai
bwa samse -n 1000000 lambda_contigs.fna lambda_trim.read2.sai \
   lambda_trim.read2.fastq.gz 2> lambda_trim.read2.samse.log > \
   lambda_trim.read2.sam
bwa aln -c -n 0.001 -l 18 lambda_contigs.fna \
   lambda_trim.single.fastq.gz > lambda_trim.single.sai
bwa samse -n 1000000 lambda_contigs.fna lambda_trim.single.sai \
   lambda_trim.single.fastq.gz 2>lambda_trim.single.samse.log > \
   lambda_trim.single.sam
```

The resulting alignment files are in SAM format, which is what the next operation requires.

C.2.4 Constructing an assembly graph

Now we will use the `ref_select` tool to convert the colorspace contig sequences to nucleotides and output the matepair assembly graph JSON file.

```
ref_select -q -m --mp_mate_cnt=10 -r lambda_trim.read1.fastq.gz -r \
   lambda_trim.read2.fastq.gz -r lambda_trim.single.fastq.gz \
   lambda_trim.read1.sam lambda_trim.read2.sam \
   lambda_trim.single.sam > lambda_asm.json
```

C.2.5 Producing scaffolded sequence

```
# The following chain of graph_ops commands accomplish the following
# steps:
#
# MST -    Convert the assembly graph into a directed spanning tree
# PLUCK -  Remove short (stub) branches from the tree, leaving a
#          mostly linear structure
# PRUNE -  Cleave any remaining branches from the tree, leaving linear
#          backbones
# PUSH -   Put back contigs removed from the ends of the scaffolds by
#          PLUCK
# INSERT - Put back small contigs that fit between the backbone
#          contigs
# SCAFF -  Re-linearize the scaffolds based on constraints added by
#          INSERT
# FASTA -  Join all neighboring contigs together (looking for
#          overlaps) and write FASTA to STDOUT

graph_ops lambda_asm.json MST PLUCK PRUNE PUSH INSERT SCAFF FASTA \
   '{"scaff":true}' > lambda_scaffs.fna
```

C.2.5.1 Visualizing the sequence graph, and using `SCRIPT` files with `graph_ops`.

If you rerun `graph_ops` using the `DOT` command at each stage of the pipeline, you can then visualize the assembly graph as it progresses through each stage of processing using Graphviz.

First, create a text file called `lambda_viz.go` with the following contents:

```
# Sample script file for SEAStAR graph_ops tool
DOT {"file":"lambda_asm#.dot"}
MST
DOT {"file":"lambda_asm#.dot"}
PLUCK
DOT {"file":"lambda_asm#.dot"}
PRUNE
DOT {"file":"lambda_asm#.dot"}
PUSH
DOT {"file":"lambda_asm#.dot"}
INSERT
DOT {"file":"lambda_asm#.dot"}
SCAFF
DOT {"file":"lambda_asm#.dot"}
FASTA {"scaff":true,"file":"lambda_scaffs.fna"}
```

Now run the following command:

```
graph_ops lambda_asm.json lambda_viz.go
```

This will produce the same scaffolded assembly as the previous example, but in addition, it will write a `DOT` (GraphViz) format file for each step in the scaffolding pipeline (with an incrementing number at the end of the filename: `lambda_asm0.dot`, `lambda_asm1.dot`, etc). This next command will convert all of these DOT files to corresponding PDF figures using the `neato` layout engine in Graphviz.

```
# After this command, load the output PDF files in a viewer to see the
# scaffold layout process, step by step. Circles are contigs with area
# proportional to sequence length and color by %GC. Black arrows are
```

```
# mate-pair connections with thickness indicating bitscore. Red arrows
# are added dependencies to produce a fully ordered layout for SCAFF.

for i in lambda_asm?.dot; do neato -Tpdf $i > ${i%.*}.pdf; done
```

You should now be able to load the PDF files with your favorite PDF viewer in order from `lambda_asm0.pdf` through `lambda_asm6.pdf` to see the effect of each `graph_ops` command in the script above. Each graph represents the state of the assembly before/after each successive step, showing the operation of graph_ops as it operated on this example dataset.

C.2.6 Estimating 16S sequence abundance

The tools in SEAStAR can be used to create a pipeline for estimating sequence abundance in a short-read data set. This is particularly useful for assessing community composition in a metagenomic sample through alignments to a 16S database. See this walk-through of an implementation of this pipeline using the RDP 16S database.

C.3 Command Reference

Tools described in this user guide:

- **`solid2fastq`** – Conversion of *SOLiD specific* CSFASTA/QUAL or FASTQ files to SEAStAR style FASTQ format files
- **`fastq_nodup`** – Reference-free PCR deduplication of paired reads
- **`samplefastq`** – Subsamples reads randomly
- **`trimfastq`** – Tunable, pairing aware read-trimmer
- **`ref_select`** – Converts SAM reference alignments into JSON "sequence graph" files for assembly and abundance analyses
- **`graph_ops`** – A toolkit for operating on the `ref_select` "sequence graph" files to produce assemblies and data tables

- **seq_scaffold** – Converts pre-ordered and oriented contigs into scaffolded sequence, filling gaps where possible
- **tetracalc** – Clusters sequence scaffolds into "species-level" taxonomic bins for final assembly
- **misc_scripts** – Miscellaneous scripts

C.4 solid2fastq

This tool converts SOLiD colorspace `.csfasta` and `.qual` input files (raw reads transferred off the instrument) to output files that are an interoperable variant of the FASTQ file format (see SEAStAR format details in section FASTQ). It can also convert nucleotide FASTQ files produced by runs using Exact Call Chemistry to SEAStAR style FASTQ files. This is conversion necessary to ensure that read pairs are synced in `.read1.fastq` and `.read2.fastq`.

C.4.1 Usage:

```
solid2fastq [options] <in_prefix> <out_prefix>
```

Where `<in_prefix>` denotes the input prefix which is the part of the input filename shared in common between all `.csfasta` and `.qual` or `.fastq` files generated by the same sequence library. It is typically shown in the `Title:` comment line near the top of `.csfasta` format input files. For example:

```
# Title: GG050409_20090604_matepair_50
```

is found near the top of input files:

```
GG050409_20090604_matepair_50_R3.csfasta
GG050409_20090604_matepair_50_R3_QV.qual
GG050409_20090604_matepair_50_F3.csfasta
GG050409_20090604_matepair_50_F3_QV.qual
```

So use of GG050409_20090604_matepair_50 for <in_prefix> will specify these four files for input to the solid2fastq tool.

Note that solid2fastq relies on the SOLiD naming conventions for the suffixes of these files (e.g. _R3.csfasta). You are free to rename these files with different prefixes, but the suffixes must remain the same. You may, however, compress these files using gzip and add the customary .gz suffix at the very end of the filename, which tells solid2fastq to decompress the files as it reads them.

The <out_prefix> parameter is similar to the <in_prefix> as it is used to name the output .fastq file(s) generated by this tool. The output prefix may be any text, except you should generally avoid whitespace and punctuation characters other than "_". This is because, by default, the names of all output reads are also prefixed with this string and the variant of the FASTQ format used by the SEAStAR tools depends on certain punctuation characters to quickly parse these read names (see SEAStAR FASTQ format). This file naming constraint may be relaxed through the use of optional parameters described below.

Note that <in_prefix> and/or <out_prefix> may contain directory path information, but in the case of <out_prefix> this will necessitate the use of the --prefix or --no_prefix options (described below).

solid2fastq writes a single output .fastq file for each matching pair of .csfasta / .qual input files or for each single .fastq input file. For example, given a set of non-barcoded colorspace input files, the names of these output files are constructed as:

```
<in_prefix>_R3[.csfasta|_QV.qual] → <out_prefix>.read1.fastq
<in_prefix>_F3[.csfasta|_QV.qual] → <out_prefix>.read2.fastq
```

singlets from F3/R3 → <out_prefix>.single.fastq

Note that for paired-end libraries, alternate naming suffix conventions for the R3 input are also recognized (substituting F5-BC, F5-P2, F5-DNA, F5-RNA, for R3).

Additionally, for paired libraries (mate-paired or paired-end) should `solid2fastq` encounter any singlet reads (those with a missing mate), they are by default given the appropriate suffix and then written to a file named: `<out_prefix>.single.fastq`

The complete list of file name patterns that will be recognized by `solid2fastq` are:

```
<in_prefix>_<BC>_R3.fastq → <out_prefix>.read1.fastq
<in_prefix>_[R3|F5-BC|F5-P2|F5-DNA|F5-RNA][.csfasta|_QV.qual|.QV.qual] →
   <out_prefix>.read1.fastq
<in_prefix>_<BC>_[R3|F5-BC|F5-P2|F5-DNA|F5-RNA][.csfasta|.QV.qual] →
   <out_prefix>.read1.fastq
<in_prefix>_[R3|F5-BC|F5-P2|F5-DNA|F5-RNA]_[<BC>.csfasta|_QV_<BC>.qual] →
   <out_prefix>.read1.fastq
<in_prefix>_<BC>_F3.fastq → <out_prefix>.read2.fastq
<in_prefix>_F3[.csfasta|_QV.qual|.QV.qual] → <out_prefix>.read2.fastq
<in_prefix>_<BC>_F3[.csfasta|.QV.qual] → <out_prefix>.read2.fastq
<in_prefix>_F3_[<BC>.csfasta|_QV_<BC>.qual] → <out_prefix>.read2.fastq
```

C.4.2 `solid2fastq` optional parameters `[options]`:

```
[-h|--help] [--version] [--prefix=<string>] [-n|--no_prefix]
[-x|--no_suffix] [-z|--gzip] [-b <BC>|--bc=<BC>] [-s|--singles]
```

`-h` / `--help`

Print a guide to valid command line parameters and their correct usage to the terminal, and then exit.

`--version`

Print the version number of the executing program and then exit.

`--prefix=<string>`

Specify the prefix string to add to read identifiers. For example, `.csfasta` read identifier: `>1_68_381_R3` will become: `@xx+<string>:1_68_381/1` in the `.fastq` output file. By default this is `<out_prefix>` described above.

The use of a terse, but unique, prefix string is advisable because it helps identify the source, and processing steps, performed on a set of reads as they flow through an analysis pipeline. Keep in mind that this text will be added to the output file for every read, so short strings are preferable. The output prefix may be any text you would like, except you may not use whitespace or punctuation characters other than "_".

`-n` / `--no_prefix`

Overrides the default read prefixing behavior and instead preserves the read identifiers as they are in the input `.csfasta` file(s).

For example: `>1_68_381_R3` simply becomes `@xx+:1_68_381/1`

`-x` / `--no_suffix`

Suppress /1 or /2 suffix additions to read IDs. This ensures matching read IDs for paired reads in FASTQ files and non-BWA aligner generated SAM files.

`-z` / `--gzip`

Write output `.fastq` files compressed in the gzip format, and with filenames suffixed `.gz`.

`-b <BC>` / `--bc=<BC>`

Used only for barcoded SOLiD libraries, in addition to `<in_prefix>`, to specify input files for a specific barcode (`<BC>`).

`-s` / `--singles`

Write two separate singlet files that segregate the singlets by input files. This is useful for doing strand-specific analyses using paired-end libraries, where the mated reads are sequenced from opposite strands. For example:

`<in_prefix>_R3[.csfasta|_QV.qual] → <out_prefix>.read1.fastq`

singlets from `R3` → `<out_prefix>.single1.fastq`

`<in_prefix>_F3[.csfasta|_QV.qual] → <out_prefix>.read2.fastq`

singlets from `F3` → `<out_prefix>.single2.fastq`

C.5 ***`fastq_nodup`***

This tool reads paired `.fastq` files (with associated singlets), and removes the lowest quality pair(s) of presumed PCR duplicate reads. This operation assumes that multiple read-pairs sharing substantially the same sequences for both mates are statistically unlikely to occur frequently at random (due to the distribution of insert sizes), and therefore represent likely artifactual duplications resulting from PCR over-amplification during library construction. If this assumption does not hold for a given library, then use of this tool is inappropriate.

Notably, this tool is most appropriate for metagenomes (or any other sample type) where no reference genome is available for performing duplicate removal through reference alignments. This is because `fastq_nodup` works entirely through error-tolerant internal comparisons of read-pairs to each other within an entire sequenced library.

To accomplish this, `fastq_nodup` builds a large table of sequence prefixes, requiring a substantial amount of memory. The amount of memory required can be controlled through tuning the various parameters, but a minimum of 32Gb will probably be necessary for most read datasets.

Note that singlet reads are also randomly removed in the same proportion as matching mated reads. This is done assuming that PCR duplicated pairs are as likely to produce singlets as non-duplicated pairs. For example:

`S1`, `S1`, `M1 -- M2`, `M1 -- M2`, `M1 -- M3`

In the above example `Sn` = singlet reads, `Mn` = mated reads, and the numeric suffixes indicate matching sequences. So the lowest quality pair of `M1 -- M2` mates will be removed from the output, and each of the `S1` reads will have a 1/3 probability of being removed

(disregarding quality). The 1/3 probability is calculated from the fact that 1/3 of the pairs containing sequence `M1` were presumed PCR duplicates and therefore removed.

C.5.1 Usage:

```
fastq_nodup [options] <in_prefix> <out_prefix>
```

where `<in_prefix>` and `<out_prefix>` specify the input and output filename prefixes to use for input files and naming output files. For example:

```
<in_prefix>.read1.fastq → <out_prefix>.read1.fastq
<in_prefix>.read2.fastq → <out_prefix>.read2.fastq
<in_prefix>.single.fastq → <out_prefix>.single.fastq
```

The input files may be gzip format compressed files with the customary `.gz` suffix at the very end of the filename, which tells `fastq_nodup` to decompress the files as it reads them. Multiple single files (i.e. `.single1.fastq` and `.single2.fastq` suffixes) are also recognized.

C.5.2 `fastq_nodup` optional parameters `[options]`:

```
[-h|--help] [--version] [-v|--verbose] [-z|--gzip]
[--prefix=<string>] [--no_prefix] [-l <u>|--index_len=<u>]
[-m <u>|--match_len=<u>] [-d <u>|--index_err=<u>]
[-e <u>|--match_err=<u>] [--seed=<n>]
```

`-h` / `--help`

Print a guide to valid command line parameters and their correct usage to the terminal, and then exit.

`--version`

Print the version number of the executing program and then exit.

`-v` / `--verbose`

Print additional statistics and diagnostic information about the run.

-z / **--gzip**

Write output `.fastq` files compressed in the gzip format, and with filenames suffixed `.gz`.

--prefix=<string>

Specify the prefix string to add to read identifiers. For example: `@<string>:1_68_381`

Note, this will replace any existing prefix. By default this is `<out_prefix>` described above.

--no_prefix

Overrides the default read prefixing behavior and instead preserves the read identifiers exactly as they are in the input fastq file(s).

-l <u> / **--index_len=<u>**

Controls the read sequence prefix length of the search index. This should be set long enough so that random prefix collisions are infrequent, but short enough so that a large number of sequence errors are unlikely to accumulate. This parameter also impacts the memory use of this tool, with each increment increasing the index size by a factor of 4. This parameter defaults to `<u>`=14 nucleotides and may not exceed 32 nucleotides.

Both mates of each read-pair are indexed, and this indexed look-up is used to find a list of mate sequences corresponding to each sequence prefix. Given that the mates also must match, it is acceptable for a low-level of prefix collisions to occur in this index. That is, assuming the level of prefix collision is low (only a small number of non-duplicated reads share any given prefix), it will be highly improbable that reads matching a given prefix will share mates with matching sequence at random.

-m <u> / **--match_len=<u>**

Controls the required match length of the unindexed mate in detecting a duplicate pair.

Like the previous parameter (`--index_len`), this setting has an impact on memory use, although memory requirements only increase linearly with match length. This parameter should be set as high as memory permits to improve specificity.

This parameter defaults to the full read length, although lower settings are acceptable assuming that the `--index_len` is appropriately set, and the sum of `--index_len` and `--match_len` exceeds about 35 nucleotides (that is, random 35 nucleotide joint index-match collisions are extremely unlikely assuming each match is sufficiently long).

`-d <u>` / `--index_err=<u>`

Controls the number of mismatches tolerated in the indexed sequence prefix while still permitting a sequence match. This value defaults to <u>= 1 and may be set in the range 0-2. This setting has a major impact on run time, with each increment increasing it by a factor of about 4. Duplicated reads with more than `--index_err` mismatches in the `--index_len` prefix may still be found by the mate's indexed prefix, assuming that it does not also have more than `--index_err` errors.

`-e <u>` / `--match_err=<u>`

Controls the number of mismatches tolerated in a matched mate sequence (in the `--match_len` prefix). This value defaults to 3 and must be > 0. This setting has little impact on run time. Setting `--match_err` to higher values will reduce the specificity of the mate-matching and should therefore be accompanied by correspondingly larger values of `--match_len`.

`-seed=<n>`

Integer seed used by the internal pseudo-random number generator. This only affects the random selection of matching singlets (as described above). Defaults to <n>= 12345.

*C.6 **samplefastq***

This tool randomly samples reads from a given set of fastq files, producing a corresponding set of output fastq files. This is useful for more quickly tuning parameters for the `trimfastq`tool (described below) and for producing simulated read datasets for various other purposes.

C.6.1 Usage:

```
samplefastq -f|--frac_sample=<float>|<int> [options] <in_prefix>
   <out_prefix>
```

where the mandatory `-f` parameter indicates how the input files should be sampled and `<in_prefix>` and `<out_prefix>` specify the input and output filename prefixes to use for input files and naming output files. For example:

```
<in_prefix>.read1.fastq → <out_prefix>.read1.fastq
<in_prefix>.read2.fastq → <out_prefix>.read2.fastq
<in_prefix>.single.fastq → <out_prefix>.single.fastq
```

The input files may be gzip format compressed files with the customary `.gz` suffix at the very end of the filename, which tells `samplefastq` to decompress the files as it reads them. Multiple single files (i.e. `.single1.fastq` and `.single2.fastq` suffixes) are also recognized.

Note that for paired sequences (as shown above) read pairs are always sampled together; that is, each pair of reads counts at a single read unit for sampling purposes, as does each singlet read.

*C.6.2 `samplefastq` mandatory parameter: **[-f|--frac_sample=<float>|<int>]***

`-f <float>` / **`-f <int>`** / **`--frac_sample=<float>`** / **`--frac_sample=<int>`**

The `-f` parameter tells `samplefastq`, either directly or indirectly, how many output reads to randomly sample from the input `.fastq` files. Values < 1.0 are interpreted as a fraction of the input reads to retain, whereas values > 1.0 are rounded to an integer count of reads to retain.

Note that when a read count is provided, the input files must be read twice by `samplefastq`; once to count the number of input reads, and then again to randomly sample those reads. Once the number of input reads is known, the requested read count is converted into a fraction and operation proceeds as though that fraction had been initially provided to the `-f` parameter. Therefore, if a given set of inputs is to be sampled repeated, it will be considerably more efficient to obtain the count of reads once and calculate the fraction to retain for each subsequent sampling. The requested (or calculated) fraction of reads to sample is used internally as the probability of retaining each read (or pair) and each is considered an independent trial. Because of this, the precise fraction (or count) of reads requested is unlikely to be generated, although the sampled output will typically be very close to that requested.

C.6.3 `samplefastq` optional parameters `[options]`:

`[-h|--help] [--version] [--seed=<int>] [-z|--gzip]`

`-h` / **`--help`**

Print a guide to valid command line parameters and their correct usage to the terminal, and then exit.

`--version`

Print the version number of the executing program and then exit.

`-z` / **`--gzip`**

Write output fastq files compressed in the gzip format, and with filenames suffixed `.gz`.

--seed=<n>

Integer seed used by the internal pseudo-random number generator. To obtain different samples from the same input file(s), set this parameter to different values. Repeated runs with all parameters the same will produce identical outputs. Defaults to <n>= 12345.

C.7 ***trimfastq***

This tool is used to trim or reject individual reads on the basis of quality scores. Reads may also be rejected based on low information content (entropy). When run on paired reads, this operation is performed on mates independently, and when one mate of a pair is rejected, the other becomes a singlet.

Trimming based on quality scores is performed with the goal of producing trimmed reads that have a calculated probability of error below some threshold. That is, bases are trimmed off the 3' end of the read until the remaining bases have a joint probability of error below a threshold. For example, if a threshold of 0.5 is selected, that means the output reads will be expected to have, on average, 0.5 erroneous nucleotides per read.

For colorspace (SOLiD differentially encoded) reads, the probability of error is calculated as the joint probability of error in each of two consecutive color positions. That is, because single color errors will typically be correctable, the probability of a nucleotide error at a given position is the probability that both corresponding color positions are erroneous (i.e. the pairwise product of color error probabilities yields the probability of uncorrectable nucleotide errors).

Minimum read lengths may also specified such that reads trimmed shorter than the specified minimum to meet an error probability threshold will instead be rejected.

Reads may also be independently rejected for failing to have a minimum mean information content. `trimfastq` optionally calculates the mean entropy of each read (as average bits per dinucleotide) and reject reads that fail to meet a given threshold. A common error for next-generation sequencers is to produce junk reads where all or part of the read contains highly repetitive (low information) sequence. Random DNA sequence contains 4 bits per dinucleotide of entropy, and we have observed that (non-repeat) natural sequences typically contain more than 3 bits per dinucleotide.

C.7.1 Usage:

```
trimfastq [options] <in_prefix> <out_prefix>
```

where `<in_prefix>` and `<out_prefix>` specify the input and output filename prefixes to use for input files and naming output files. For example:

```
<in_prefix>.read1.fastq → <out_prefix>.read1.fastq
<in_prefix>.read2.fastq → <out_prefix>.read2.fastq
<in_prefix>.single.fastq → <out_prefix>.single.fastq
```

The input files may be gzip format compressed files with the customary `.gz` suffix at the very end of the filename, which tells `trimfastq` to decompress the files as it reads them. Multiple single files (i.e. `.single1.fastq` and `.single2.fastq` suffixes) are also recognized. It is also possible to write FASTA format files (i.e. lacking quality scores) and interleaved paired-read files (for the Velvet assembler). See the options below for more information.

C.7.2 `trimfastq` optional parameters `[options]`:

```
[-h|--help] [--version] [-c|--color_space] [-z|--gzip]
[-v|--invert_singles] [-s|--singles] [-p <d>|--correct_prob=<d>]
[-l <u>|--min_read_len=<u>] [-m <u>|--min_mate_len=<u>]
[-f <u>|--fixed_len=<u>] [--prefix=<string>] [--add_len]
[--no_prefix] [-e <d>|--entropy_filter=<d>] [--entropy_strict]
```

```
[--mates_file] [--no_rev] [--only_mates] [--fasta]
```

-h / **--help**

Print a guide to valid command line parameters and their correct usage to the terminal, and then exit.

--version

Print the version number of the executing program and then exit.

-z / **--gzip**

Write output fastq files compressed in the gzip format, and with filenames suffixed ‘.gz+.

--prefix=<string>

Specify the prefix string to add to read identifiers. For example: `@<string>:1_68_381`

Note, this will replace any existing prefix. By default this is `<out_prefix>` described above.

-n / **--no_prefix**

Overrides the default read prefixing behavior and instead preserves the read identifiers exactly as they are in the input fastq file(s).

--add_len

Add the final trimmed length value to the read prefix. For example:

```
@lambda:1_81_912
TGTTGTGCGCGCCATAGCGAGGGGCTCAGCACGCGTCCCTCCGCCCCAC
+
5/0)358)%57*)%881/(/4'.'53*&#91*-'&,&%$''%&'46-+#`
```

Trims to: (“37” in the first line indicates the final trimmed length)

```
@37|lambda_trim:1_81_912
TGTTGTGCGCGCCATAGCGAGGGGCTCAGCACGCGTC
+
5/0)358)%57*)%881/(/4'.'53*&#91*-'&,&
```

-v / **--invert_singles**

Causes singlet file(s) output to be the opposite configuration of the input. That is:

`<in_prefix>.single.fastq → <out_prefix>.single[12].fastq`

or

`<in_prefix>.single[12].fastq → <out_prefix>.single.fastq`

Note that this option is only valid when there are input singlet reads.

-s / **--singles**

Write two singlet files (`.single1.fastq` and `.single2.fastq`), one for new singlets generated from each paired input file. Note that this option is only valid when there are no input singlet reads. The default behavior in this case is to write a single combined singlet file (`.single.fastq`).

-p <d> / **--correct_prob=<d>**

Minimum mean probability that trimmed output reads are free of uncorrectable errors (or all errors with `-c`). Default value is <d>= 0.5 Setting `-p` to 0.0 completely disables quality trimming, which is sometimes appropriate for file format conversion and interleaving operations (e.g. see `--fasta` and `--mates_file`).

-l <u> / **--min_read_len=<u>**

Minimum length of a singlet or longest-mate of a pair, in nucleotides. Default value is <u>= 24 bases.

-m <u> / **--min_mate_len=<u>**

Minimum length of the shortest mate of a pair, in nucleotides. Default value is the value provided for `--min_read_len`. This value must be < the value used for `--min_read_len`. Use of this parameter allows paired reads shorter than `--min_read_len` to be retained as long as their mate satisfies `--min_read_len`.

-f <u> / --fixed_len=<u>

Trim all reads to a fixed length, still rejecting reads that don't meet specified quality thresholds at this length. Default is no fixed length.

-e <d> / --entropy_filter=<d>

Reject reads with mean per position measured information (entropy) below the given value (in bits per dinucleotide). The range of valid values is 0.0-4.0 inclusive. By default this filter is off.

--entropy_strict

Reject reads for low entropy overall, not just the retained part after trimming. By default this setting is off. Use of this setting requires use of `--entropy_filter`.

--mates_file

In addition to other outputs, produce a Velvet compatible interleaved paired output file (named `<out_prefix>_mates.fastq`) with read2 mates reversed by default (to support SOLiD mate-paired reads).

--no_rev

By default, for SOLiD colorspace read-pairs, the second read of each pair is reversed in the `mates` file produced by `--mates_file` (to put the mates on opposite strands, as Velvet expects). `--no_rev` disables this reversing, which is useful for SOLiD colorspace paired-end files. Nucleotide inputs are by default not reversed in the output generated by `--mates_file`, so this option has no effect in that case.

--force_rev

By default, for nucleotide (non-colorspace) read-pairs, the second read of each pair isn't reversed in the `mates` file produced by `--mates_file` (to keep the mates on opposite strands, as Velvet expects, assuming Illumina sequence). `--force_rev` enables this reversing, which is useful for SOLiD 5500+ nucleotide mate-paired files (nucleotide sequences are correctly

reverse complemented). Colorspace inputs are reversed by default in the output generated by `--mates_file`, so this option has no effect in that case.

`--only_mates`

Surpress writing `<out_prefix>.read1.fastq` and `<out_prefix>.read2.fastq` output files. `--only_mates` may only be used with the `--mates_file` option. This option is useful when reads are being trimmed exclusively for assembly, to save unnecessary processing for files that will not be used. See also the`--fasta` option below to save even more space.

`--fasta`

Write FASTA format files instead of FASTQ files for all outputs. FASTA files are about half the size of FASTQ, so for some purposes (e.g. assembly, alignment) that may not take quality scores into account, FASTA may be a preferable output format to save disk space. When `--fasta` is used, no `.fastq` files will be written, and the output files will instead end with `.fasta[.gz]`.

C.8 `ref_select`

`ref_select` is a utility for calculating a wide variety of useful information from alignments of next-generation sequence reads to large reference sequence datasets. For example, it can be used to build a meta-genomic assembly graph from paired-reads aligned back to *de novo* assembled contigs. It can also be used to precisely select (and accurately estimate the relative abundances of) sequences from an alignment with a reference library (e.g. metagenomic reads to 16S rDNA database, or mRNA transcript reads to a database of genes from one or more genomes).

`ref_select` processes SAM format alignment files into a JSON format "sequence graph" file ready to be processed by the `graph_ops` tool. Optionally, `ref_select` will also read

FASTQ files and filter out reads (and mates) which are or aren't present in the provided SAM alignment. The JSON output file is structured as a graph of "nodes" (selected sequences) and "edges" (connections between nodes based on read-pairing) A variety of useful information for the nodes and edges is included in the output JSON file, which can be accessed using the `graph_ops` tool or custom code written in any language with a JSON library.

C.8.1 Usage:

```
ref_select [options] <sam_file1> [<sam_file2>]... > <outfile.json>
```

`ref_select` has a lot of options, and depending on what type of analysis you are doing, you may use quite a few of them. However, at its simplest, `ref_select` reads one or more SAM alignment files and writes a JSON "sequence graph" to STDOUT:

```
ref_select alignment.single.sam > seq_graph.json
```

`ref_select` requires any SAM (or FASTQ) format input files to be named using the SEAStAR naming convention used for FASTQ files. Crucially, individual SAM alignment files must not mix alignments for reads from different categories (`read1`, `read2`, `single`), and the corresponding filename suffixes of the SAM files must match those of the FASTQ files used to generate them:

```
sample.read1.fastq → alignment.read1.sam
sample.read2.fastq → alignment.read2.sam
sample.single.fastq → alignment.single.sam
```

Note that `ref_select` may also write warnings, errors and diagnostic information to STDERR, so it is important to keep STDOUT and STDERR separate when saving the output to a file, or the resulting saved JSON data file syntax may be corrupted by messages written to STDERR.

Of course, since these types of files may be very large, you may prefer to keep everything compressed:

```
ref_select alignment.single.sam.gz | gzip -c > seq_graph.json.gz
```

Because `ref_select` has many options, and produces many different kinds of outputs, we made the decision to organise all of the output data into a single output stream in a standard text-based format (JSON), that is flexible and straightforward to process in any language. The JSON formatted output produced by `ref_select` is documented in the appendix: JSON Sequence Graph File Format Reference. However, for most tasks you will not need to look directly into these files because the `graph_ops` tool will handle that work for you (including the work of extracting data into other standard file formats).

`ref_select` has four main pieces of functionality:

1. Quantitatively analyzing reads aligned to a known reference database (e.g. metagenomic reads to a 16S rDNA database, or reads from mRNA transcripts to the gene models of one or more genomes):

   ```
   # Select only reference sequences with reads aligned scoring
   # 100 or more bits
   ref_select -t 100.0 alignment.single.sam > ref_stats.json

   # Convert ref_select output to a tab separated (TSV) table for
   # use downstream
   graph_ops ref_stats.json TABLE > ref_stats.tsv
   ```

2. Analyzing reads mapped back to *de novo* assembled contigs, to build a "sequence graph" for downstream quality checking, scaffolding, binning, visualization, etc.

   ```
   # Construct a sequence graph from the alignment, inserting the
   # assembled contig seqs into the graph
   ```

```
ref_select --ref=contigs.fna -m alignment.read1.sam \
   alignment.read2.sam > seq_graph.json

# Produce scaffolded sequence (where .go file is graph_ops
# SCRIPT)...
graph_ops seq_graph.json run_scaffold_pipeline.go > \
   scaffolds.fna
```

3. Converting SOLiD colorspace contigs into nucleotide contigs based on aligned reads

```
# Rather than inserting contigs as above, they are
# reconstructed from the alignment and the provided read FASTQ
# files
ref_select -q -m -r mates.read1.fastq -r mates.read2.fastq \
   alignment.read1.sam alignment.read2.sam > seq_graph.json

# Produce scaffolded sequence (where .go file is graph_ops
# SCRIPT)...
graph_ops seq_graph.json run_scaffold_pipeline.go > \
   scaffolds.fna
```

4. Filtering out a sub-set of reads (and their mates) based on the results of an alignment.

```
# Write two new read files for all reads (and their mates)
# that did not map in the alignment
# NOTE: this is the one exception to the "all data goes into
# the JSON stream" principle described above.
# Filtered read output data is written to `FASTQ` format
# files, with a prefix specified using --read_output
ref_select --read_output=filtered_ --output_nomatch -r \
   mates.read1.fastq -r mates.read2.fastq \
   alignment.read1.sam alignment.read2.sam
```

The approaches used by `ref_select` to accomplish these tasks are described in more detail in the methods section of [64] (Iverson, *et al.* 2012).

C.8.2 The `[options]` reference below organizes the various parameters into general categories related to the above tasks:

- Basic parameters
- Bitscore calculation and reference selection parameters
- Coverage calculation parameters
- Sequence reconstruction and read filtering parameters
- Read pairing statistical parameters
- Input file parameters

C.8.2.1 `ref_select` basic parameters `[options]`:

```
[-h|--help] [-v|--version] [--verbose]
[-d <seq_id>|--detail=<seq_id>]... [--detail_file=<detail_file>]
[-q|--recon_seq] [--ref=<ref_file>] [-m|--mate_pair]
[--split=<n>] [--separate_strands]  [--rollup] [--seed=<n>]
[--num_threads=<n>]
```

`-h` / `--help`

Request help.

`-v` / `--version`

Print the build version and exit.

`--verbose`

Print detailed diagnostic status to stderr during run. Off by default.

`-d <seq_id>` / `--detail=<seq_id>`

Only produce per-base statistics and reconstructed sequence for listed seq ids. By default these things are done for all selected sequences. For very large reference sequence databases, `ref_select` needs to perform a lot of work and can use a lot of memory and produce a very large output. If you want to analyze the entire reference dataset, but only produce

detailed results for a small subset of sequences, the `--detail` option allows you to specify just those sequences (by `<seq_id>`, one per use of `-d` / `--detail`). To specify more than a few sequences, use `--detail_file` instead.

`--detail_file=<detail_file>`

Filename of file containing sequence ids to treat as in `--detail`. Format is a plain text file with one `<seq_id>` per line.

`-q` / `--recon_seq`

Enable reconstruction of contig sequences using alignment data (SAM) and reads (FASTQ). This works for both colorspace and nucleotide data (or a combination of the two), and the resulting sequence is always nucleotide sequence. The reconstructed contigs may contain ambiguity codes (e.g. for SNPs or variable positions) and the sensitivity of this process is controlled with the `ambig_tol` parameter. NOTES: Requires use of `--read_file`. If `--detail` is used, only specified sequences are reconstructed.

`--ref=<ref_file>`

For assemblies where the contigs and alignments are already nucleotide sequences, this option allows you to specify a FASTA file containing the reference contigs used in the alignment (as an alternative to using `--recon_seq`). In this case, the reference sequences are simply copies into the output JSON sequence graph for use in downstream analysis (scaffolding, binning, etc).

`-m` / `--mate_pair`

This option is required to enable the pairing analysis between mated reads (`.read1` and `.read2` alignments). When paired reads are available and `--mate_pair` is used, "edges" will be included in the output JSON data file, connecting the "nodes" into a full sequence graph structure. Enable this option when you want to produce scaffolds for a *de novo* assembly.

`--split=<n>`

Split reference sequences into n-base pieces for analysis. The default is no splitting. This is a specialized option that performs all `ref_select` analyses as though large reference sequences (e.g. whole genomes) are actually split into many separate chunks. This can be useful for testing, but also for performing genome connection and rearrangement analyses when pairing information is available and the sequence read data is from a different organism (e.g. strain or closely related species) than the reference sequence used.

`--separate_strands`

Top and bottom stands of reference sequences are considered separately. Each reference sequence effectively becomes two references, with reads mapping to each strand contributing to the analysis for that strand's reference only. NOTE: `--separate_strands` may not be used with: mate-pairing (`--mate_pair`), sequence reconstruction (`--recon_seq`) or splitting (`--split`)

`--rollup`

When `--split` is used, or when a `--catalog` file is specified (e.g. to join reference scaffolds or chromosomes into whole genomes), the output JSON data will include a `rollup_stats` object with information about each parent reference sequence (e.g. whole genome) calculated from the stats of the individual underlying sequences.

`--seed=<n>`

SEAStAR uses its own random number generator (for reproducibility among different platforms). You may use a different seed to investigate what difference randomized decisions made in the analysis are having on the results. By default `--seed=12345`.

`--num_threads=<n>`

`ref_select` is highly multithreaded to support modern multi-core processors. Sometimes you will want to restrict the number of cores it uses (e.g. to prevent resource competition on clusters or shared computers, or when you are running more than one instance

of `ref_select` at a time on a given computer). By default `--num_threads` is the number of cores on the machine (including "hyper-threads"). The minimum number of threads required for proper operation is 2 or 3 depending on the parmeters being used, and ref_select checks to ensure that the proper minimum number of threads is allocated.

C.8.2.2 `ref_select` parameters for bitscore calculation and reference selection `[options]`:

```
[-t <n>|--bit_thresh=<n>] [-f <n>|--bit_fraction=<n>]
[-l <n>|--read_map_limit=<n>] [-s|--second_chance_reads]
[--relax_read_sharing] [--all_taxa] [-a|--absolute_bitscores]
[--no_rand] [--sim_frac=<n>] [-e <seq_id>|--exclude=<seq_id>]...
[--exclude_file=<exclude_file>] [--invert_exclude]
```

`-t <n>` / `--bit_thresh=<n>`

Minimum bitscore value for a ref sequence to be selected for output. Default > 0.0, all sequences with a positive bitscore. NOTE: if both `--bit_fraction` and `--bit_thresh` are used, the highest resulting threshold will be used. Use of selection thresholds is useful for selecting a "noise floor" to exclude sequences that have only a small number of reads uniquely aligning, which in a large sequence database (e.g. 16S rDNA databases) may only reflect sequencing errors.

`-f <n>` / `--bit_fraction=<n>`

Minimum bitscore value for a ref sequence to be selected for output, as a fraction of the bitscore for the top scoring sequence. NOTE: if both `--bit_fraction` and `--bit_thresh` are used, the highest resulting threshold will be used.

`-l <n>` / `--read_map_limit=<n>`

Maximum number of reference mappings before a read is rejected from consideration. Default is 50000 reference sequences. Reads that map with a large fraction of the sequences in a reference database (e.g. low complexity reads, or highly conserved sequence) carry very

little information but require a lot of computational effort and memory to process. If your sequence database has a lot of conserved or redundant sequence (e.g. 16S rDNA), then this parameter will save a lot of computational expense with little if any effect on the results. If desired, a 2-pass approach can be used where reads are aligned to the full database and run through `ref_select` first to eliminate the vast majority of the database from consideration. Then in a second pass, reads are re-aligned to just the sequences selected in the first round, so that all aligning reads can be accounted for (e.g. in coverage and relative abundance calculations).

`-s` / `--second_chance_reads`

Allow reads to be mapped to edit dist +1 mappings when their best ref is eliminated. By default, if a read maps to a sequence that ultimately doesn't meet the bitscore threshold being used limit selection, that read is removed from further downstream consideration, if it didn't also map at least as well with some other sequence that is ultimately selected for output. When `--second_chance_reads` is used, such reads gain a "second chance" to be reallocated to reference sequence(s) where they map with edit distance +1 from the eliminated best match(es). This parameter causes `ref_select` to use significantly more memory. Note: this has no effect when `--bit_fraction` and `--bit_thresh` are at their default values, or when `--relax_read_sharing` is used.

`--relax_read_sharing`

Allow reads to be mapped to all sequences, regardless of edit distance. By default, reads are only associated with the sequence(s) that they align with at the minimum edit distance. `--relax_read_sharing` causes reads to be associated with all sequence alignments, even if they are suboptimal relative to the best match.

`--all_taxa`

Use count of all taxa in catalog to calculate bitscores. By default, any taxa in the reference

database that have zero read alignments are excluded from consideration in the information theoretic bitscore calculations (ie. the calculations proceed as though those sequences didn't exist in the reference database at all).

`-a` / `--absolute_bitscores`

Use absolute bitscores to score and select reference sequences. By default length normalized scores (bits/nt) are used to control for differences in reference sequence length.

`--no_rand`

By default, reads that map equally well to multiple locations within a single reference sequence are randomly assigned to one of the possible positions. The `--no_rand` option changes this behavior so that such a read will be mapped to the first position in the reference seen in the alignment file. Counterintuitively, it is likely that using `--no_rand` will make the output of `ref_select` ***less*** deterministic. This is because the pseudo-random number generator used by SEAStAR is cross-platform deterministic, whereas the process by which such multiple alignments are ordered in an alignment SAM file may be at least paritally random (say through multi-thread race contingencies in the alignment software).

`--sim_frac=<n>`

Fraction of aligned reads to use. Reads (and their mates) are randomly sampled from the input alignment(s) in this proportion. Primarily for use with simulated datasets.

`-e <seq_id>` / `--exclude=<seq_id>`

This option in conceptually similar to the `--detail` option, except that reference sequences identified with `--exclude` are discarded from the analysis entirely, as though they were not in the reference database to begin with. This is useful if a long, expensive alignment to a large reference sequence database has been completed, and then some downstream analysis calls for processing alignments with only a subset of that database.

`--exclude_file=<exclude_file>`

Filename of file containing sequence ids to treat as in `--exclude`. Format is a plain text file with one `<seq_id>` per line.

`--invert_exclude`

`--exclude` or `--exclude_file` sequences are inverted; that is, exclude all sequences except those specified.

C.8.2.3 `ref_select` parameters for coverage calculations `[options]`:

```
[--per_base] [-w <n>|--cov_window=<n>] [--gc_window=<n>]
[--detect_dups]
```

`--per_base`

Per base statistics (coverage; and when `--mate_pair` is enabled, physical coverage and mean insert length) will be generated and output for all selected reference sequences (or only those specified by `--detail`).

`-w <n>` / `--cov_window=<n>`

Fixed read length for use in calculating per position coverage. By default the actual length of each read is used when available in the FASTQ readnames (see the SEAStAR FASTQ format guide for details). If the readnames do not include length information and `--cov_window` isn't specified, then 49 nucleotides is the default fixed read length used.

`--gc_window=<n>`

Enables per position %GC moving average calculation, within the specified window size. The results are output in the `per_nt_mean_gc` array in the JSON output for each selected node (or only those specified by `--detail`). Use of this option requires `--per_base` statistics and either `--recon_seq` or `--ref` to provide the sequence data nrcessary for the calculation. If `--mp_circular` is used, then the %GC calculation wraps around at the sequence ends with no discontinuity, otherwise the N/2 %GC values at each end of the sequence all have the same value as the first (or last) valid window.

`--detect_dups`

Perform additional analysis to detect likely collapsed duplications. Assumes relatively uniform coverage (i.e. a single isolate genome). Results are reported in the `contig_problems` section of the JSON sequence graph, a summary of which is available via the `graph_ops` `PROBS` command.

C.8.2.4 `ref_select` parameters for sequence reconstruction and read filtering `[options]`:

```
[--ambig_tol=<n>] [--read_output=<prefix>] [--output_nomatch]
[--read_output_gzip]
```

`--ambig_tol=<n>`

Tolerance adjusting the sensitivity for generating ambiguity codes in output sequence reconstructed from read alignments. In the range: 0.0 - 1.0. 1.0 = no ambiguities, majority rules base calling. Default is 0.2

`--read_output=<prefix>`

Enables output of reads aligning with the refernce database (and their mates, regardless of whether the mates align). `<prefix>` indicates the FASTQ output filename prefix to use in writing the new ouput files. Must be used with `--read_file` and/or `rev_read_file`.

`--output_nomatch`

Inverse the meaning `--read_output` to write only reads that do not align with any reference sequence (whose mates also do not align).

`--read_output_gzip`

Modifies `--read_output` to write gzip compressed output files.

C.8.2.5 `ref_select` parameters for generating read pairing statistics `[options]`:

```
[--mp_mate_lim=<n>] [--mp_share_lim=<n>] [--mp_strict]
[--mp_inserts] [--mp_circular] [--mp_cutoff=<n>]
```

--mp_mate_cnt=<n>

Minimum count of mapping read-pairs for generation of a mate-pair linking edge in the output graph (normal or internal). Default is 2, meaning that no edges consisting of a single read-pair will be output.

--mp_mate_lim=<n>

Minimum bitscore threshold for generation of a mate-pair linking edge in the output graph (normal or internal). Default is 0.0 bits, however `--mp_mate_count` above will put a floor on lowest scoring edges output.

--mp_share_lim=<n>

Minimum bitscore threshold for generation of a shared sequence edge in the output graph. Default is 500.0 bits.

--mp_strict

Do not consider paired reads that span reference sequences when the same pair of reads also map entirely within some single reference sequence.

--mp_inserts

Perform insert size estimation for all reference sequences where at least one pair of reads both map within that sequence.

--mp_circular

Allow circular self-linking mate-pairs to join the opposite ends of a single sequence.

--mp_cutoff=<n>

Insert size cutoff for inclusion of an individual mate-pair in per base statistics. Setting this option for some reasonable upper limit of the length of a valid insert will prevent sequence duplications from improperly skewing estimation of the true mean insert size.

C.8.2.6 `ref_select` parameters for specifying input files `[options]`:

```
[-r <read_file>|--read_file=<read_file>]...
[--rev_read_file=<read_file>]... [--old_bwa_samse]
[-c <catalog_file>|--catalog=<catalog_file>]
[--rev_align=<sam_file>]...
```

NOTE! all inputs to `ref_select` involving per read data (FASTQ and SAM files) must be named using the SEAStAR FASTQ naming conventions, also see the introductory description of ref_select for an example.

`-r <read_file>` / `--read_file=<read_file>`

Input filename(s) for FASTQ read files used for sequence reconstruction and/or read filtering.

`--rev_read_file=<read_file>`

Same as `--read_file` above, but used for paired reads that are on the opposite strand from each other. By default, `ref_select` assumes that paired reads are from the same DNA strand. When this is not true, then one of the two readsets (read1/read2) must be specified using `--rev_read_file`. Note, this must be done consistently with the use of `--rev_align`.

`--old_bwa_samse`

Use the old BWA-style "samse" alignment format instead of SAM. This format was used by the BWA `samse -n` command until BWA version 0.5.6.

`-c <catalog_file>` / `--catalog=<catalog_file>`

Filename of a reference catalog file. Required for use `--old_bwa_samse`. This is optional otherwise, although it is required to associate sequence descriptions with reference sequences in the JSON output, or for use of the `--rollup` functionality when reference sequences are actually subsequences of parent sequences (e.g. scaffolds or chromosomes of a whole genome). A catalog is a tab separated (TSV) plain text file with the following columns:

```
seq_id     seq_len     parent_id   seq_description
```

For example:

```
ABCD123    1102938     ORG1        Organism 1, Scaffold 1
EFGH987    482273      ORG1        Organism 1, Scaffold 2
ZYXW765    193475      ORG2        Organism 2, Scaffold 1
RTSP345    782273      ORG2        Organism 2, Scaffold 2
```

Note that each `seq_id` must be unique, but duplicated `parent_id` values indicate relationships among sequences (a shared parent sequence).

`--rev_align=<sam_file>`

Same as the main `<sam_file>` input filenames, but used for paired reads that are on the opposite strand from each other. By default, `ref_select` assumes that paired reads are from the same DNA strand. When this is not true, then alignments for one of the two readsets must be specified using `--rev_align`. Note, this must be done consistently, and also with the use of `--rev_read_file` when input FASTQ files are also used.

C.9 ***`graph_ops`***

`graph_ops` reads JSON sequence graph files produced by the `ref_select` program. It implements a variety of commands for manipulating this data for assembly, visualization or output to a variety of file formats (as detailed for each subcommand below). Because the JSON sequence graph objects are often very large, and therefore may take significant time to read and write, we designed `graph_ops` as a federated suite of tools that can pass the current sequence graph state from one tool to the next in-memory. This saves considerable time and I/O traffic relative to an alternative design that used separate command line tools for each operation performed by `graph_ops`. A consequence of this design is that `graph_ops`

may be used in either a scripted mode, where all commands are supplied via the command-line or a script (`.go`) file; or in an interactive mode, where commands are manually entered one at a time.

C.9.1 Usage:

```
graph_ops [-h|--help] [<input.json>] [<script.go>]
   [<command> ['{params}']]...
```

where:

- **`-h`** / **`--help`** provides command line help regarding program usage and version information

  ```
  # Provide help about shell command line usage
  graph_ops --help
  # Provide help about graph_ops commands
  graph_ops HELP   # HELP is actually a graph_ops command
  ```

- **`<command>`** is some valid command and **`'{params}'`** optionally specifies parameters for each command, in the form: `'{"parm1":value1,"parm2":value2...}'`. Multiple commands with optional parameters may be provided in succession for execution. NOTE: the parameters for each command must be single quoted to ensure that it is passed to `graph_ops` as a single item, and to prevent interactions with special characters used by the UNIX shell.

  ```
  # Read a sequence graph and output all of the sequences to a
  # FASTA file
  graph_ops LOAD '{"file","assembly.json"}' FASTA \
     '{"file":"sequence.fna"}'
  ```

Like most of the other tools in SEAtAR, `graph_ops` can read and write gzip compressed versions of all of its input and output files:

```
# Read a compressed sequence graph and output all of the
# sequences to a compressed FASTA file
graph_ops LOAD '{"file","assembly.json.gz"}' FASTA \
   '{"file":"sequence.fna.gz"}'
```

- **<input.json>** is an optional shortcut specifying a sequence graph file to initially `LOAD`

```
# Load the sequence graph file and go interactive
graph_ops assembly.json

# Equivalent to
graph_ops LOAD '{"file":"assembly.json"}' SCRIPT
```

- **<script.go>** is an optional `SCRIPT` to initially execute

```
# Load the sequence graph file and run the scaffold script
graph_ops assembly.json write_fasta.go

# Equivalent to
graph_ops LOAD '{"file":"assembly.json"}' SCRIPT \
   '{"file":"write_fasta.go"}'

# If the write_fasta script contains the LOAD command, then
# just this will suffice
graph_ops write_fasta.go
```

In this last case, the `write_fasta.go` script could be (note that single quotes around the parameters in a `.go` file are not necessary):

```
# This is the write_fasta.go file
LOAD {"file":"assembly.json"}
FASTA {"file":"sequence.fna"}
```

Running `graph_ops` without ***any*** options will launch in interactive mode, equivalent to:

```
graph_ops SCRIPT
```

Note that many `graph_ops` commands may write warnings, errors and diagnostic information to STDERR, so it is important to keep STDOUT and STDERR separate when using commands such as `DUMP`, `DOT`, `TABLE`, or `FASTA` (without specifying an output filename) and redirecting the STDOUT of `graph_ops` to a file. Likewise, only one such output command may use STDOUT per run of `graph_ops` (e.g. redirecting the output of both `FASTA` and `DUMP` to the same file will result in a file that is both invalid JSON and invalid FASTA format).

C.9.2 `graph_ops <command>` summary:

C.9.2.1 File I/O commands

- **`LOAD`** – Input current data structure from a file, or by default from STDIN
- **`DUMP`** – Output current data structure to a file, or by default to STDOUT
- **`TABLE`** – Write `ref_select` statistics for selected reference sequences to a TSV file
- **`FASTA`** – Write sequences contained in the selected graph to the FASTA format
- **`DOT`** – Write the current assembly graph to the graphviz DOT format

C.9.2.2 Assembly pipeline commands (in this order)

- **`MST`** – Calculate the Maximal Spanning Tree of all connected components
- **`SST`** – Calculate the improved Scaffold Spanning Tree of all connected components. `SST` is an optional replacement for `MST`, useful for metagenomes or relatively short contigs.
- **`PLUCK`** – Remove all leaf contig nodes (in or outdegree == 0) from the graph
- **`PRUNE`** – Split the assembly graph at all contig nodes with in or out degree > 1
- **`SLICE`** – Break linear scaffolds at a GC / coverage discontinuity. `SLICE` is optional, but recommended for metagenomes.
- **`PUSH`** – Extends scaffold ends (reversing the action of PLUCK at scaffold ends)

- **INSERT** – Reconnect contigs with ambiguous placement within a scaffold using mapped pair position information to resolve ambiguities
- **SCAFF** – Lay out a fully linear scaffold from contigs in unambiguously ordered connected components
- **CLUST** – Cluster scaffolds using the `tetracalc` tool

C.9.2.3 Assembly graph filter / edit utilities

- **CIRCLE** – Find circular sequences among scaffolds.
- **CCOMPS** – Calculate the current graph connected components
- **SELCC** – Select specific connected components for further processing
- **SELCLUST** – Select clusters of scaffolds for further processing
- **SELND** – Select contigs from the connected neighborhoods of the given contigs
- **SCAFLNK** – Find edge connections linking scaffold ends
- **RELINK** – Reconnect previously removed edge connections between contigs
- **EDGFLT** – Remove all edges scoring less than some number of bits
- **PROBS** – Generate a report of contigs with likely internal assembly issues
- **EDIT** – Make manual explicit edits to the selected graph structure
- **CUTND** – Create a new node from an existing node using the given coordinates

C.9.2.4 Information and control

- **GC** – Generate statistics about the assembly graph
- **GCC** – Generate statistics about the assembly graph, with details about each connected component
- **STASH** – Copy (push) the current graph to the top of a stack in memory
- **UNSTASH** – Restore (pop) the current graph from the top of the stack (undo changes)

- **SCRIPT** – Use contents of file as a series of commands, or read from the console if no file provided
- **HELP** – List all valid commands, or provide detailed help for a specific command with `HELP <COMMAND>`

C.9.3 `graph_ops` detailed `<command>` reference:

*C.9.3.1 **LOAD***

Input JSON sequence graph data from a file, or by default from STDIN.

On the command line, if the first parameter isn't a valid command string, and it ends in `.json[.gz]`, then it is assumed to be the name of a JSON file, and an implicit `LOAD` command will be run using that filename.

Parameters:

- **`file : "filename.json[.gz]"`**

  Specify the name of a JSON format sequence graph file.

  Examples:

  ```
  # Read data from the file my_assembly.json
  LOAD {"file":"my_assembly.json"}

  # Read data piped from STDIN (default)
  LOAD {"file":"-"}
  ```

- **`files : ["filename1.json[.gz]",...]`**

  Specify the names of multiple JSON format sequence graph files to merge together.

NOTE: The files must have uniquely named sequence nodes (contigs) with no nodes in common among the multiple files being loaded. Only the processing history from the first JSON file is retained (the other histories may still be found in their original files). Other processing output such as scaffolds and clusters are lost.

Example:

```
# Read and merge data from the files my_assembly_1.json and
# my_assembly_2.json
LOAD {"files":["my_assembly_1.json","my_assembly_2.json"]}
```

- **edges : "edge_file.json[.gz]"**

Specify the name of a JSON format sequence graph file to load edges and mate-pair statistics from.

This option works with the `file` option, and requires that the edge file contain a graph with a subset of the nodes of the main graph being loaded with `file`. Any edges and mate-pair statisics present in the main graph are discarded in favor of those from the `edge` file.

Example:

```
# Read nodes and sequence data from main_file.json, and edges
# and connection data from edge_file.json
LOAD {"file":"main_file.json","edge_file":"edge_file.json"}
```

C.9.3.2 ***DUMP***

Output current sequence graph data to a JSON file, or by default to STDOUT

Parameters:

- **file : "filename.json[.gz]"**

Specify a file name to write the JSON format sequence graph to. If the filename contains one or more # characters in a row, these positions are replaced with zero-padded digits that

will increment each time a file is written to this filename pattern. If no # characters are present, then this command overwrites any existing file of the same name.

Examples:

```
# Write data to the file my_assembly.json
DUMP {"file":"my_assembly.json"}

# Write data piped to STDOUT (default)
DUMP {"file":"-"}

# Write data to the file: my_assembly_000.json (first time
# this is run)
DUMP {"file":"my_assembly_###.json"}

# Run again, write data to the file: my_assembly_001.json
DUMP {"file":"my_assembly_###.json"}
```

C.9.3.3 ***TABLE***

Write `ref_select` statistics for selected reference sequences to a TSV file. The fields are (left to right by column):

- `bitscore` - Information content of reads aligning with the ref sequence
- `read_cnt` - Number of (possibly fractional) reads aligning with the ref sequence
- `norm_cnt` - Read_cnt normalized to ref sequence length
- `rel_abun` - Relative fractional abundance of (copy number of) the ref sequence to all selected sequences (those with bitscores above some thresh)
- `mean_cov` - Mean coverage of the reference sequence by (possibly fractional) reads
- `read_len` - Mean length of reads aligning with this sequence
- `seq_len` - Length of the ref sequence
- `pct_gc` - Percent GC content of ref sequence (NA if not calculated)

- `name` - Catalog name of the ref sequence
- `desc` - Catalog description of the ref sequence

Parameters:

- `file : "filename.tsv[.gz]"`

Specify a file name to write the TSV format stats table to. If the filename contains one or more # characters in a row, these positions are replaced with zero-padded digits that will increment each time a file is written to this filename pattern. If no # characters are present, then this command overwrites any existing file of the same name. (See `DUMP` command an example using this behavior)

Examples:

```
# Write stats to the file my_seq.tsv
TABLE {"file":"my_seq.tsv"}

# Write stats to STDOUT (default)
TABLE {"file":"-"}
```

- `header : true`

Write a header row in the output table with labels for each column.

Example:

```
# Write a header row (false is default).
TABLE {"header":true}
```

C.9.3.4 ***FASTA***

Write sequences contained in the current graph data to a FASTA format file

Parameters:

- `file : "filename.fasta[.gz]"`

Specify a file name to write the FASTA format sequence data to. If the filename contains one or more # characters in a row, these positions are replaced with zero-padded digits that will increment each time a file is written to this filename pattern. If no # characters are present, then this command overwrites any existing file of the same name. (See `DUMP` command an example using this behavior)

Examples:

```
# Write stats to the file my_seq.fasta
FASTA {"file":"my_seq.fasta"}

# Write sequence to STDOUT (default)
FASTA {"file":"-"}
```

- `scaff : true` or `scaff : "[options]"`

Output fully scaffolded sequences (using the `seq_scaffold` tool). Using this parameter with the value `true` run `seq_scaffold` with its default settings. As an advanced option, this parameter can also accept a string argument, which will be passed along to the external `seq_scaffold` tool as its `[options]` parameter string. Run with option string `"--help"` for help with the settings offered by `seq_scaffold` and its default values.

Examples:

```
# Output scaffolded contig sequences using default settings.
FASTA {"scaff":true}

# Get seq_scaffold help
FASTA {"scaff":"--help"}

# Pass seq_scaffold some options
FASTA {"scaff":"--overlap=7 --heal=othercontigs.fna"}
```

- `no_merge_scaffs : true`

Write contig sequences in scaffold order with a scaffold ID in each header, ready to be provided to the `seq_scaffold` tool (but do not run it).

Example:

```
# Output ordered contig sequences
FASTA {"no_merge_scaffs":true}
```

- `abundance : true`

Append relative abundance values to the FASTA sequence IDs. This is for use with downstream classification and visualization tools, such as for the 16S rDNA analysis pipeline vignette.

Example:

```
# Attach abundance values
FASTA {"abundance":true} -- Attach abundances
```

C.9.3.5 ***DOT***

Write the current graph data to a graphviz format DOT file for visualization or other processing. Some of the parameters listed below allow you to change node and edge parameters that will affect the output style of graphs rendered using the `dot` or `neato` layout tools that are part of the graphviz package. The parameters provided here give only very limited control of the most common style choices available. The `gvpr` language supplied by graphviz is much better suited for more detailed manipulation of the myriad of rendering options available.

Parameters:

- `file : "filename.dot[.gz]"`

Specify a file name to write the DOT format graph data to. If the filename contains one or more # characters in a row, these positions are replaced with zero-padded digits that will increment each time a file is written to this filename pattern. If no # characters are present, then this command overwrites any existing file of the same name. (See `DUMP` command an example using this behavior)

Example:

```
# Write stats to the file my_graph.dot
DOT {"file":"my_graph.dot"}

# Write graph to STDOUT (default)
DOT {"file":"-"}
```

- `detail : true`

Draw labelled contig nodes

Example:

```
# Draw contigs as ovals containing the node names of each
# contig. Note that this disables scaling the node size by
# contig length.
DOT {"detail":true}
```

- `arrowtype : "arrowtype_string"`

Control the type of arrowheads drawn. See graphviz documentation at: http://www.graphviz.org/doc/info/attrs.html#k:arrowType

Example:

```
# Draw normal arrows (default)
DOT {"arrowtype":"normal"}
```

- `pen_scale : <float>`

Control the relative thickness of the arrow lines. See graphviz documentation at: http://www.graphviz.org/doc/info/attrs.html#d:penwidth

Example:

```
# Draw normal arrows (default)
DOT {"pen_scale":0.1}
```

- `const_edge : true`

Draw connection arrow lines at constant width

Example:

```
# Draw edge arrows of constant width regardless of bitscore.
DOT {"const_edge":true}
```

- `colored_edges : true`

Draw edges colored by the GC% of the connected contigs

Example:

```
# Draw colored arrows
DOT {"colored_edges":true}
```

C.9.3.6 ***MST***

Calculate the Maximal Spanning Tree of all connected components. This command reduces the connection graph to a tree-structure that optimally maximizes the total remaining edge bitscores, preserving all previously connected components of the graph, and directs all remaining edges of the graph based on the pairing information.

Parameters:

- `bits : true`

Use raw connection bitscores instead the default heuristically adjusted scores (based on GC% / coverage differences)

Example:

```
# Use raw connection bitscores from mate-pairing
MST {"bits":true}
```

C.9.3.7 ***SST***

As an alternative to the `MST` command, calculate the Scaffold Spanning Tree of all connected components, producing a sub-optimal tree from the perspective of maximizing edge bitscores, but with important properties for successful contig scaffolding in the following steps. This command is generally preferable to the MST command when median contig length is less than the mean distance between paired reads. That is, when a relatively large mate-pair insert size was selected for the sequencing library, relative to the size of the contigs that could ultimately be assembled. This condition is often the case for metagenomic samples, where inter-strain variability adversely affects the median contig length that can be achieved.

Parameters:

- `bits : true`

Use raw connection bitscores instead the default heuristically adjusted scores (based on GC% / coverage differences)

Example:

```
# Use raw connection bitscores from mate-pairing
SST {"bits":true}
```

C.9.3.8 ***PLUCK***

Remove all leaf contig nodes (in- or out-degree == 0) from the (tree-structured) graph. Two (or sometimes three) iterations are usually sufficient to achieve the goal of removing all non-terminal short branches from the tree. These contigs will be added back to the assembly in later operations once the "backbone" scaffolds have been determined. NOTE: `PLUCK` does not remove two "leaf" contig nodes that form a connected pair; that is, multiple iterations of `PLUCK` will not completely remove contigs that happen to form short chains. Also, `PLUCK` preserves entirely unconnected contigs with more than 20000 bases of sequence by default. See the `min_len` parameter below.

Parameters:

- `iterate : <int>`

Specify the number of iterations of PLUCK to run. Each is equivalent to running PLUCK again as a separate command. By default `<int>`= 3.

Examples:

```
# Default is {"iterate":2}
PLUCK

# This is the equivalent of the above command
PLUCK {"iterate":1}
PLUCK {"iterate":1}
```

- `min_len : <int>`

Minimum length of isolated nodes to keep. Default `<int>`= 20000 bases.

Example:

```
# Do not "pluck away" an unconnected contig containing 20000
# or more bases of sequence.
PLUCK  {"min_len":20000}
```

C.9.3.9 ***PRUNE***

Split the assembly graph at all contig nodes with in or out degree > 1. This command prunes remaining branches from the assembly tree by selectively removing the lowest scoring extra in- or out- edges from a node. After this operation is complete, all remaining connected components will be linear and properly directed.

Parameters:

- `strict : true`

  Always cut all but the strongest link when there are more than two in + out links

  Examples:

  ```
  # The default case: attempt to determine which branch(es) to
  # remove in order to preserve one high weight path across a
  # multiply linked contig node.
  PRUNE {"strict":false}

  # Always strictly prune. Equivalent to {"ratio":0.0} below
  PRUNE {"strict":true}
  ```

- `ratio : <float>`

  Ratio of the strongest to next strongest links to trigger strict pruning

  Examples:

  ```
  # Always strictly prune.
  PRUNE {"ratio":0.0}

  # Never strictly prune.
  PRUNE {"ratio":1.0}

  # Default. Don't strictly filter when the score of the highest
  # scoring edge is >= 5x greater than the next highest scoring
  # link of the same direction (for both in and outlinks).
  PRUNE {"ratio":0.2}
  ```

- `verbose : true`

  Output diagnostics on STDERR

  Example:

```
# Generate extra output information
PRUNE {"verbose":true}
```

C.9.3.10 ***SLICE***

Break linear scaffolds at a %GC / Coverage discontinuity. Each run of `SLICE` will break a given scaffold at no more than one position. `SLICE` should be run multiple times if multiple misassemblies per scaffold are suspected.

Parameters:

- `thresh : <float>`

  Threshold used to determine whether or not to break a connection

  Example:

```
# Default. Medium strength heuristic score based on GC% /
# Coverage statistics of a scaffold on either side of a given
# contig's connection edge. The lower the threshold, the more
# sensitive the filter is to such discontinuities.
SLICE '{"thresh":0.5}'
```

C.9.3.11 ***PUSH***

Extends scaffold ends, reversing the (undesired) action of `PLUCK` at scaffold ends. Because the `PLUCK` command can't distinguish short-branches from what will ultimately become scaffold ends, it essentially erodes a contig off of the each end with every iteration. To reverse this, `PUSH` should be run the same number of iterations as was `PLUCK` previously.

Parameters:

- `iterate : <int>`

Number of iterations of PUSH to run. Each is equivalent to running `PUSH` again as a separate command. By default `<int>`= 3.

Examples:

```
# Default case.
PUSH {"iterate":3}

# Equivalent to above.
PUSH {"iterate":2}
PUSH {"iterate":1}
```

C.9.3.12 ***INSERT***

Reconnect contigs with ambiguous placement within a scaffold using relative mean read-pair mapping position information to resolve ambiguities. By default this will run on all scaffolds. `INSERT` works by fully connecting contigs within the scaffold when their relative position can be unambiguously determined, and adding new "constraint" edges (supported only by the relative mapped position information) between such contigs so that they can be ordered correctly by the `SCAFF` command.

Parameters:

- `ccname : "contig_name"`

Use only the connected component (scaffold) containing the named contig.

Example:

```
# Insert removed contigs within the connected component
# containing this contig
INSERT {"ccname":"NODE_1234"}
```

- `ccnames : ["contig_name1","contig_name2",...]`

Insert removed contigs within only the connected components (scaffolds) containing these contigs.

Example:

```
INSERT {"ccnames":["NODE_1234","NODE_567"]}
```

- `thresh : <float>`

Minimum contig connection bitscore to consider

Example:

```
# Default. Only connections scoring at least 250.0 bits
# will count in the calculations
INSERT {"thresh":250.0}
```

- `min_pos_diff : <int>`

Minimum mapping position difference (within a neighboring contig) considered reliable enough to use for resolving positional ambiguities.

Example:

```
# Default. Only pairing information yielding a relative
# position difference of 75 nucleotides (between candidate
# contigs and a neighboring existing contig) will be
# considered significant enough to use in determining the
# relative ordering of the candidate contigs
INSERT {"min_pos_diff":75}
```

- `dup_kmer : <int>`

Length of k-mers to use for detecting variant duplicate contigs

Examples:

```
# Default. Use 14-mers to detect duplicate variant contigs
# before inserting them between scaffold backbone contigs.
INSERT {"dup_kmer":14}

# Disable duplicate variant contig checks.
INSERT {"dup_kmer":0}
```

- `dup_thresh : <float>`

Fraction of kmers from a contig with hits to scaffold backbone.

Example:

```
# Default. If more than 15% of kmers for a contig hit kmers
# from the scaffold backbone contigs, then do not insert that
# contig.
INSERT {"dup_thresh":0.15}
```

- `external : true`

Controls whether contigs may be inserted outside of an input scaffold.

Examples:

```
# Default.
# Contigs will not be added outside of the first and last
# contigs in the input scaffold; that is, the only inserts
# performed will be between existing connected contigs.
INSERT {"external":false}

# Contigs may be added to the scaffold before the first, or
# after the last, contig in an input scaffold.
INSERT {"external":true}
```

- `verbose : true`

Output diagnostics on STDERR

Example:

```
# Generate extra output information
INSERT {"verbose":true}
```

C.9.3.13 ***SCAFF***

Lay out a fully linear scaffold from contigs in unambiguously ordered connected components (as produced by the `INSERT` command)

Parameters: NONE.

C.9.3.14 ***CLUST***

Cluster scaffolds using the external `tetracalc` tool

Parameters:

- `options : <string>`

Command line options for the `tetracalc` tool, otherwise tool defaults used. Run with `--help` for help with the settings offered by `tetracalc`.

Examples:

```
CLUST {"options":"--merge_tar=0.95 -m 7500"}

CLUST {"options":"--help"}

CLUST {"options":"--fixed -t 0.9 -s 0.8 -r 0.9"}
```

C.9.3.15 ***CIRCLE***

Find circular sequences among scaffolds

Parameters:

- `circular : true`

Return circularized scaffolds.

Default (`false`): Produce linear scaffolds placing the ends within the largest contig of each connected component. The node with the longest sequence will be divided equally into two and the mate-pair connections also divided, such that they form two clean ends for the new scaffold.

Example:

```
# Produce circular scaffolds.
CIRCLE {"circular":true}
```

- `thresh : <float>`

Minimum bitscore of end-connecting pairing information

Example:

```
# Default. Circle completing end connections must have a
# positive bit score to qualify.
CIRCLE {"thresh":0.0}
```

C.9.3.16 ***CCOMPS***

Calculate the current assembly graph connected components. Note, this is a low-level operation which is typically done automatically by other commands as required.

Parameters:

- `sortby : <string>`

Specify how to sort the resulting list of CCs. Must be: “nodes” or “sequences”.

Examples:

```
# Default. Sort CCs in descending order of number of nodes.
CCOMPS {"sortby":"nodes"}

# Sort CCs in descending order of amount of sequence.
CCOMPS {"sortby":"sequences"}
```

C.9.3.17 ***SELCC***

Select specific connected components for further processing. "Unselected" connected components are saved in a "removed" data structure and may be recalled with other commands.

NOTE: There are two types of parameters below. "Selection parameters" allow a subset of connected components to be selected by contig names or component number. "Filter parameters" are applied to the set of "selected" connected components (or by default, to the set of all connected components). These filters may also be used in combination, resulting in a logical "AND" relationship (ie. connected components not satisfying all of the filters are removed).

Selection Parameters:

- `ccname : "contig_name"`

  Select the connected component containing the named contig

  Example:

```
# Select the connected component containing the contig named
# NODE_1234
SELCC {"ccname":"NODE_1234"}
```

- `ccnames : ["contig_name1","contig_name2",...]` Select the connected component(s) containing the named contigs

  Example:

```
# Select the connected components containing the contigs named
# NODE_1234 and NODE_5678
SELCC {"ccnames":["NODE_1234","NODE_5678"]}
```

- `ccnum : <int>`

Select connected component number <int>

Example:

```
# Default. Select connected component number 0 (the first CC).
SELCC {"ccnum":0}
```

- ccnums : [<int>,<int>,...]

Select the connected components from the list of numbers

Example:

```
# Select the second and third connected components (numbering
# is zero based)
SELCC {"ccnums":[1,2]}
```

- ccrange : [<int1>,<int2>]

Select the connected components numbered in the range <int1>..<int2> (inclusive). <int> may be negative, indicating positions at the end of the list of connected components.

Examples:

```
# Select the first 6 connected components
SELCC {"ccrange":[0,5]}

# Select the last 5 connected components
SELCC {"ccrange":[-5,-1]}
```

- shift : true

Select all connected components except the first one. This is like SELCC {"ccrange":[1,-1]} except it doesn't generate an error when there is only one remaining connected component, allowing processing to potentially continue in any calling SCRIPT commands.

Example:

```
# Drop the first connected component.
SELCC {"shift":true}
```

Filter Parameters:

- `min_nodes : <int>`

Select connected components with or more nodes.

Example:

```
# Select connected components with 2 or more nodes.
SELCC {"min_nodes":2}
```

- `min_seqlen : <int>`

Select connected components with or more sequence within nodes.

Example:

```
# Select connected components containing at least 1000 bases
# of sequence.
SELCC {"min_seqlen":1000}
```

- `sequence : <string>`

Select connected components containing the provided DNA sequence. NOTE! This isn't BLAST, the sequence must match exactly. Any differences, including ambiguity codes, will prevent matching. The only extra thing that is done is the reverse complement of the provided sequence is also searched.

Example:

```
# Select connected components containing the provided sequence
# (or its reverse complement).
SELCC {"sequence":"AGACTAGCAGATATACGATAACGATACGATACGAT"}
```

- `sequences : [<string>,...]`

Like `sequence` parameter above, but using a list of sequences

Example:

```
# Select connected components containing the provided
# sequences (or their reverse complements).
SELCC {"sequences":["AGACTAGCAGATATACG","ATAACGATACGATACGAT"]}
```

- `soft : true`

Determines whether unselected ccomp nodes are deleted or kept as "removed"

Examples:

```
# The contigs of first connected component will remain
# selected, all others are retained as unselected.
SELCC {"soft":true}

# The contigs of first connected component will remain
# selected, neighboring contigs (one hop away) are unselected
# and added to the "removed" pool. All others are permanently
# deleted.
SELCC {"soft":false}  # Default
```

C.9.3.18 ***SELCLUST***

Select clusters of scaffolds for further processing

Parameters:

- `clustnum : <int>`

Select a specific cluster

Example:

```
# select the first cluster of scaffolds
SELCLUST {"clustnum":0}
```

- `clustnums : [<int1>, <int2>, ...]`

Select multiple specific clusters

Example:

```
# Select these three clusters
SELCLUST {"clustnums":[0,3,5]}
```

- `clustrange : [<int1>, <int2>]`

Select a range of clusters

Examples:

```
# Select the first six clusters
SELCLUST {"clustrange":[0,5]}

# Select the last five clusters
SELCLUST '{"clustrange":[-5,-1]}'

# select all clusters except the last one
SELCLUST '{"clustrange":[0,-2]}'
```

- `shift : true`

Select all clusters except the first one. This is like `SELCLUST {"clustrange":[1,-1]}` except it doesn't generate an error when there is only one remaining cluster, allowing processing to potentially continue in any calling `SCRIPT` commands.

Example:

```
# Drop the first cluster.
SELCLUST {"shift":true}
```

- `exclusive : true`

  Remove all scaffolds outside of the selected clusters

  Example:

  ```
  # Default. Keep all unselected scaffolds for subsequence
  # processing
  SELCLUST '{"exclusive":false}'
  ```

C.9.3.19 ***SELND***

Select contigs from the connected neighborhood(s) of the given contig(s)

Parameters:

- `name : "contig_name"`

  Use a single contig by name.

  Example:

  ```
  # Select NODE_1234 (and its neighbors)
  SELND {"name":"NODE_1234"}
  ```

- `names : ["contig_name1","contig_name2",...]`

  Use multiple contigs by name.

  Example:

  ```
  # Select these two contigs (and their neighbors...)
  SELND {"names":["NODE_1234","NODE_5678"]}
  ```

- `radius : <int>`

  Size of the neighborhood of contigs to select

  Examples:

```
# Select all neighbors and neighbors of neighbors
SELND {"radius":2} --

# Default. Select only the named contig(s)
SELND {"radius":0}
```

- `sequence : <string>`

Select contig(s) found to contain the provided DNA sequence.

NOTE! This isn't BLAST, the sequence must match exactly. Any differences, including ambiguity codes, etc. will prevent matching. The only extra thing that is done is the reverse complement of the provided sequence is also searched.

Example:

```
# Select contig(s) containing the provided sequence (or its
# reverse complement).
SELND {"sequence":"AGACTAGCAGATATACGATAACGATACGATACGAT"}
```

- `sequences : [<string>,...]`

Like `sequence` parameter above, but using a list of sequences

Example:

```
# Select contigs containing the provided sequences (or their
# reverse complements).
SELCC {"sequences":["AGACTAGCAGATATACG","ATAACGATACGATACGAT"]}
```

C.9.3.20 ***SCAFLNK***

Find connections linking scaffold ends

Parameters:

- `scaff_names : ["Scaffold_name1","Scaffold_name2",...]`

Select a subset of scaffolds to attempt to link to other scaffolds. Default: use all scaffolds.

Example:

```
# Select only the scaffolds named Scaffold_1234 and
# Scaffold_5678 for processing. Only scaffold links involving
# one of these scaffolds will be considered.
SCAFLNK {"scaff_names":["Scaffold_1234","Scaffold_5678"]}
```

- `exclusive : true`

Only look for connections within the `scaff_names` list. Default: look for connections between those in `scaff_names` and all other scaffolds.

Example:

```
# Only look for connections between Scaffold_1234 and
# Scaffold_5678. All other scaffolds are ignored.
SCAFLNK {"exclusive":true, "scaff_names": ["Scaffold_1234", \
   "Scaffold_5678"]}
```

- `thresh : <float>`

Bitscore threshold

Example:

```
# Default. Only reconnect scaffold ends with connections
# scoring at least 1000.0 bits
SCAFLNK {"thresh":1000.0}
```

C.9.3.21 ***RELINK***

Reconnect previously removed connections between contigs. `RELINK` can move contigs from out of the "unselected" pool.

Parameters:

• `name : "contig_name"`

Use a single contig by name.

Example:

```
# Restore removed edges to this contig (reconnecting with the
# contigs on the other side of the edges)
RELINK {"name":"NODE_1234"}
```

• `names : ["contig_name1","contig_name2",...]`

Use multiple contigs by name.

Example:

```
# Restore edges to these two contigs (though not necessarily
# between them...)
RELINK {"names":["NODE_1234","NODE_5678"]}
```

• `ccname : "contig_name"`

Use all contigs in the connected component containing the named contig.

Example:

```
# Restore edges to all contigs in the connected component
# containing this contig
RELINK {"ccname":"NODE_1234"}
```

• `ccnames : ["contig_name1","contig_name2",...]`

Restore edges to all contigs in the connected components containing these contigs.

Example:

```
# Restore edges to all contigs within the connected
# compontent(s) containing these two contigs
RELINK {"ccnames":["NODE_1234","NODE_5678"]}
```

• radius : <int>

Expand the sphere of restored connections to neighbors.

Examples:

```
# Default value. Restore edges only to neighboring contigs
RELINK {"radius":0}

# Recursively restore edges to all contigs within two hops of
# any selected contigs (including along newly restored edge
# paths)
RELINK {"radius":2}
```

• complete : true

Restore connections among all types of contigs.

Examples:

```
# Restore connections among contigs which are both removed
# from, and part of, currently selected connected components.
# This is like running with both modes of the 'existing'
# parameter (below) simultaneously.
RELINK {"complete":true}

# Default case.
# The 'existing' parameter (below) determines which type of
# connections are restored.
RELINK {"complete":false}
```

• existing : true

Only restore connections between nodes currently part of selected connected components. This parameter has no effect when the 'complete' parmeter (above) is true.

Examples:

```
# Only restore connections between currently selected contigs
RELINK {"existing":true}

# Default case.
# Only restore connections between currently selected and
# currently unselected contigs (this causes the unselected
# contigs to become part of the selected set again). That is,
# no new connections within the contigs of currently selected
# connected components are restored.
RELINK {"existing":false}
```

- `problems : true`

Restore all connections to contigs that are marked with potential assembly problems. Uses `radius:1` and `existing:true`

Example:

```
# Reconnect all problem contigs to all of their potential
# neighbors.
RELINK {"problems":true}
```

C.9.3.22 ***EDGFLT***

Remove all edges scoring less than thresh bits

Parameters:

- `thresh : <float>`

Bitscore threshold which must be met to keep an edge.

Example:

```
# Default. Remove all edges scoring less than 500.0 bits.
EDGFLT {"thresh":500.0}
```

C.9.3.23 ***PROBS***

Generate a report of contigs with likely internal assembly issues from the `contig_problems` section of the JSON graph file. This information is optionally generated by using the `ref_select --detect_dups` parameter.

Parameters:

- `file : "filename.txt[.gz]"`

  Name of txt format file to write statistics to

  Examples:

  ```
  # Write report to the file problems.txt
  PROBS {"file":"problems.txt"}

  # Default. Write report to STDOUT
  PROBS {"file":"-"}
  ```

- `detail : true`

  Provide extra per contig detail

  Example:

  ```
  PROBS {"detail":true}
  ```

C.9.3.24 ***EDIT***

Make manual explicit edits to the selected graph structure. Using this function is preferable to directly editing the JSON structure because it provides a direct record of what was done, and makes it easily repeatable in case changes to upstream analysis are made. If more than one node-independent sets of edges are to be added or removed (that is, those not sharing any contig(s) in common), then multiple calls to `EDIT` are required to accomplish this task.

Parameters:

- `rem_nodes : ["contig1","contig2",...]`

Remove the specified contig nodes by name

Example:

```
# Remove these two contigs from the graph
EDIT {"rem_nodes":["NODE_1234","NODE_5678"]}
```

- `add_nodes : ["contig1","contig2",...]`

Add back the specified contig nodes by name

Example:

```
# Move these two contigs from the "unselected" pool back to
# the selected graph
EDIT {"add_nodes":["NODE_1234","NODE_5678"]}
```

- `rem_edges : ["contig1",["contig2",...]]`

Remove the specified connections between contig1 and any number of other contigs.

Example:

```
# Remove two connection edges from the graph, both connected
# to NODE_1234
EDIT {"rem_edges":["NODE_1234",["NODE_5678","NODE_9"]]}
```

- `add_edges : ["contig1",["contig2",...]]`

Add back the specified connections between contig1 and any number of others.

Example:

```
# Add two connection edges from the graph, both connected to
# NODE_1234
EDIT {"add_edges":["NODE_1234",["NODE_5678","NODE_9"]]}
```

NOTE: for `rem_edges` and `add_edges` above, if only a single edge is involved, then the parameter syntax may optionally be flattened.

For example:

```
EDIT {"add_edges":["NODE_1234",["NODE_9"]]}  # is equivalent to
EDIT {"add_edges":["NODE_1234","NODE_9"]}
```

C.9.3.25 ***CUTND***

Create a new node from an existing node using the given coordinates. Using this function is preferable to directly editing the JSON structure because it provides a direct record of what was done, and makes it easily repeatable in case changes to upstream analysis are made.

Parameters:

- `name : "contig_name"`

  Name of the contig node to be copied

  Example:

  ```
  # Use sequence from NODE_1234
  CUTND {"name":"NODE_1234"}
  ```

- `new_name : "new_contig_name"`

  Name for the newly created contig node

  Example:

  ```
  # New node will be named NODE_1234a
  CUTND {"new_name":"NODE_1234a"}
  ```

- `include : ["contig_name1","contig_name2",...]`

Move mate-pair edges from the listed contig(s) to the newly created "cut" version of the named contig.

Example:

```
# Edges between NODE_1234 and the "included" contigs will be
# moved to NEW_1234.
CUTND {"name":"NODE_1234", "new_name":"NEW_1234", \
   "include":["NODE_567", "NODE_234"]}
```

- `begin : <int>` and `end : <int>`

`begin` is the beginning sequence coordinate for the new node (from within the original). `end` is the ending sequence coordinate for the new node (from within the original).

NOTE: To facilitate trimming sequences to a fixed maximum length, it is allowable for negative `begin` values and positive `end` values to be longer than a sequence. In such cases, the values are set to the beginning and end of the sequence, respectively. Positive `begin` and negative `end` positions must fall within the sequence, otherwise the `begin` position will be after the `end` position.

Deprecated with SEAStAR version v0.4.13: `start` may be used as a synonym for `begin`.

Examples:

```
# The new node will include sequence from positions 123 to 456
CUTND {"begin":123,"end":456}

# The new node will include sequence from position 0 (implied)
# to 456. end may also be omitted, implying the last position
CUTND {"end":456}

# The new node will include at most the last 1000 bases of
# sequence. If the sequence is shorter than 1000 bases, it
# will be copied without modification.
CUTND {"begin":-1000}
```

C.9.3.26 ***GC***

Generate a report of statistics about the current assembly graph

Parameters:

- `file : "filename.txt[.gz]"`

Examples:

```
# Write stats to the file my_graph.txt
GC {"file":"my_graph.txt"}

# Default. Write stats to STDOUT
GC {"file":"-"}
```

- `ccdetail : true`

Write connected component and cluster details. Note the `GCC` command is shorthand for this.

Example:

```
GC {"ccdetail":true}   # Equivalent to GCC
```

C.9.3.27 ***GCC***

Generate a report of detailed statistics about the current assembly graph, with details about each connected component and scaffold cluster (if present)

Parameters:

- `file : "filename.txt[.gz]"`

Filename for writing statistics report

Examples:

```
#  Write stats to the file my_graph_ccs.txt
GCC {"file":"my_graph_ccs.txt"}

# Default. Write stats to STDOUT
GCC {"file":"-"}
```

C.9.3.28 ***STASH***

Copy (push) the current graph to the top of a stack in memory. This allows you to quickly save the current graph state (without erasing any previous "stashes") so that operations can later be undone with the `UNSTASH` command. WARNING! Each time the `STASH` command is run (without a corresponding `UNSTASH`) a **new** copy of the current sequence graph data is made and stored in memory. This is convenient, but can add up quickly and rapidly lead to out-of-memory errors and other such problems if not managed carefully.

Parameters: None.

C.9.3.29 ***UNSTASH***

Restore (pop) the current graph from the top of the stack (undo changes since most recent `STASH`). This command removes (and frees) this state from the stack while restoring it.

Parameters:

- `free : true`

Remove the stashed data from the top of the stack without restoring it (do not replace current state)

Examples:

```
STASH
# do something destructive here
UNSTASH
```

```
# undo destructive action...

STASH
# do something destructive here
UNSTASH {"free":true}
# keep effects of destructive action, but free up the STASHed state
```

C.9.3.30 ***SCRIPT***

Use contents of a file as a series of `graph_ops` commands to run. Commands and their parameters are specified one per line. Lines begging If no file is provided, enter interactive mode, reading commands one at a time from a command prompt. Note that while any extension may be used with the `file` option below, the extension `.go` is required if you wish for the script file to be recognized directly by `graph_ops` at the command line:

```
graph_ops write_fasta.go   # Execute commands in write_fasta.go

# This doesn't require ".go"
graph_ops SCRIPT '{"file":"anything.txt"}'

graph_ops SCRIPT   # Enter an interactive prompt environment

graph_ops   # Same as above
```

Parameters:

- `file : "filename.go[.gz]"`

Read commands from the named file. A `.go` file is a plain text format file with one valid `graph_ops` command and its parameters per line. Blanks lines and other whitespace are not significant. Comments may be added by prefacing them with #. Note that unlike on the UNIX command line, it is unnecessary to single quote the command parameters in a `.go` file.

```
# This is the write_fasta.go file
LOAD { "file" : "assembly.json" }

# scripts can use SCRIPT (with a file)
SCRIPT { "file" : "do_something.go" }

FASTA { "file" : "sequence.fna" }
```

- `self : true`

When used with a SCRIPT command inside a SCRIPT file, causes that script to call itself.

```
# When called inside the script file "file.go"
SCRIPT { "self" : true }
# Is exactly equivalent to:
SCRIPT { "file" : "file.go" }
# The only difference is that if file.go is renamed, the "self"
# version will continue to work as expected
```

- `tag : <string>`

Substitute the provided string in place of the '@' symbol in any output file names used by commands (e.g. `DUMP`) within the provided `SCRIPT` file. NOTE: this tag renaming is not used within interactive SCRIPT command sessions.

```
# Add the string "Run1" in place of the '@' character in the
# filenames of any output files written by commands in the
# script my_script.go (e.g. "@_output.json" will become
# "Run1_output.json" in a DUMP command). Works with any command
# that can write its output to a file.

SCRIPT {"file" : "my_script.go", "tag" : "Run1"}
```

The tag string will also substitute for runs of consecutive '@' characters, and for multiple distinct positions anywhere in an output file path. Note that as with all `graph_ops` output filenames, any resulting directory path must already exist. `graph_ops` will not create new directories in the course of writing output files.

```
# String \"Sample_A\" will replace all runs of '@' characters
# anywhere in the file path of files written by commands in
# the script my_script.go, for example:
# ~/data/samples/@@@/@@@_output.json
# will become:
# ~/data/samples/Sample_A/Sample_A_output.json

# NOTE! In this case, the ~/data/samples/Sample_A/ directory
# must already exist.

SCRIPT {"file" : "my_script.go", "tag" : "Sample_A"}
```

Note that the `SCRIPT` command and the associated `.go` scripts are not intended to replace the UNIX shell. Notably, the scripts are entirely declarative, there are no general variables, or ability to branch (if/then/else) or explicitly loop. However, using a combination of `graph_ops` commands, it is still possible to do some unexpectedly powerful things very efficiently:

```
# This script is called: iterate.go
STASH
SELCC
FASTA {"file":"@_ccseq_####.fna"}
UNSTASH
SELCC {"shift":true}
SCRIPT {"file":"iterate.go"}
```

If the above script is invoked using this command:

```
graph_ops assembly.json CCOMPS SCRIPT '{"file":"iterate.go", "tag":"Run1"}'
```

`graph_ops` will read in `assembly.json`, calculate the connected components of the sequence graph once, and then write the sequence from each individual connected component to a separate FASTA file, with output filenames automatically incrementing from `Run1_ccseq_0000.fna` to `Run1_ccseq_0234.fna` (assuming there are 235 connected components in the assembly).

C.9.3.31 ***HELP***

List all valid commands, or provide detailed help for a specific command with `HELP <COMMAND>`. The `HELP` command when used as the first and only command on the command-line (or within an interactive `SCRIPT` session) may include the topic command directly as the second command line parameter rather than using the `topic : "command"` parameter described below. For example:

```
graph_ops HELP GCC

# Is shorthand for:

graph_ops HELP '{"topic" : "GCC"}'
```

Parameters:

- `topic : "command"`

Provide detailed help for a specific command. Using the special word `ALL` (which is not a valid command) in place of the command string will cause detailed help to be printed for *all* commands.

Examples:

```
# Provide help about the HELP command itself
HELP {"topic":"HELP"}

# Provide a dump of detailed help about all commands
HELP {"topic":"ALL"}
```

- `cmd_line : true`

Provide detailed help for UNIX shell command line use. This help can also be obtained from the shell by running with the `-h` or `--help` options.

Example:

```
# UNIX command line usage help.
HELP {"cmd_line":"true"}

# Prints the same information as (at the command line):
graph_ops -h
# OR
graph_ops --help
```

C.10 ***seq_scaffold***

This program reads a nucleotide FASTA file with already properly oriented and ordered contig sequences (in scaffold order, as produced by the `graph_ops FASTA` command run with the parameter setting `"no_merge_scaffs":true`), and produces a FASTA format output with the contigs joined into scaffolds. **NOTE:** this command can be run automatically (with default settings) from within the `graph_ops FASTA` command, using the parameter setting `"scaff":true`).

In accomplishing this task, this tool attempts to do three main things:

1. Look for short sequence overlaps at the ends of neighboring contigs, while possibly disregarding a few potentially erroneous bases at the ends of either contig, in order to close the gap between the contigs. For example:

```
AGATACATCTACGCGATCGACGATCGATCCCATCGAcgt
                        aCGATCCCATCGATCGAAGTCGCGCGCATGCAGACTACGC
```

2. Optionally, use alternatively assembled contigs (e.g. generated using a different *de novo* assembler, or with different settings) in an attempt to "heal" the remaining gaps between contigs that persist after step 1. For example, the middle sequence below represents part of a contig from an alternate assembly that bridges a gap between neighboring contigs:

```
ATCTACGCGATCGACGATCGATCCCATCGA
         ...GACGATCGATCCCATCGAAGACATACCGAAGTCGCGCGCATGC...
             ^ Alternative assembly contig
                                    CGAAGTCGCGCGCATGCAGACTACGC
```

3. Where a gap cannot be bridged, fill it with a sequence of `n` (unknown bases), with the sequence length possibly adjusted (+1, or +2) to preserve the longest open reading frame that could potentially span the gap (i.e. avoid adding an artifactual frame-shift). Resulting gene models that span gaps should be thoroughly checked for occasional improper gene fusions.

```
AGATACATCTACGCGATCGAC
                     nnnnnnnnnnnnnnn  <- Possibly pad with 1 or 2 more
                                    TCGAAGTCGCGCGCATGCAGACTACGC
```

C.10.1 Usage:

```
seq_scaffold [options] <infile.fna> > <outfile.fna>
```

where `<infile.fna>` is the scaffold ordered contigs and `<outfile.fna>` is the output scaffolds. All sequence files are in FASTA format, and the input files may be compressed in the `gzip` format if the file extension ends with `.gz`. The output is written to `STDOUT`, and so may be saved to a file via redirection or piped to another command (such as `gzip`). To further facilitate pipelining, `<infile.fna>` may instead be supplied from `STDIN` by substituting "`-`" for the filename. Some examples:

```
# Simplest case, run with default settings
seq_scaffold contigs.fna > scaffolds.fna

# gzipped inputs and outputs
seq_scaffold contigs.fna.gz alt_contigs.fna.gz | gzip -c > \
   scaffolds.fna.gz

# Piped input from compressed bzip2 file
```

```
bunzip2 -c contigs.fna.bz2 | seq_scaffold - > scaffolds.fna
```

The FASTA format for the scaffold ordered contigs follows a simple convention:

```
>contig1 scaffold_name1
ATAGACGA....
>contig2 scaffold_name1
GTCAGCTC....
>contig3 scaffold_name2
CGAGCAAC....
>contig4 scaffold_name2
TTAGCGAC....
```

The above example shows an input FASTA file with two scaffolds of two contigs each. The output of `seq_scaffold` for this file will be the two joined scaffolds:

```
>scaffold_name1
ATAGACGA....GTCAGCTC....
>scaffold_name2
CGAGCAAC....TTAGCGAC....
```

In `<healfile.fna>` the sequences do not need to be in any particular order and the FASTA header sequence identifiers do not need to be unique. Note, that it is possible to simultaneously use more than one alternate assembly for "healing" by simply concatenating those assemblies into a single file.

Note that `seq_scaffold` may also write warnings, errors and diagnostic information to STDERR, so it is important to keep STDOUT and STDERR separate when saving the output to a file, or the resulting saved FASTA data may be corrupted by messages written to STDERR.

C.10.2 `seq_scaffold` optional parameters `[options]`:

```
[-h|--help|] [--overlap=<n>] [--trim=<n>] [--max=<n>]
[--heal=<healfile.fna>] [--heal_overlap=<n>] [--heal_trim=<n>]
[--heal_max=<n>] [--verbose]
```

-h / **--help**

Print a guide to valid command line parameters and their correct usage to the terminal, and then exit. Also prints the version string for this tool.

--overlap=<n>

<n> is the minimum number of bases of overlap required to join contigs.

--max=<n>

<n> is the maximum number of bases of overlap to look for.

--trim=<n>

<n> is the maximum number of non-ambiguous bases to try trimming from contig ends when looking for overlap.

--heal=<healfile.fna>

Use alternatively assembled contigs in the referenced FASTA file to try to "heal" gaps between neighboring contigs.

--heal_overlap=<n>

<n> is the minimum number of bases of overlap required to join contigs.

--heal_max=<n>

<n> is the maximum number of bases of gap to try to heal.

--heal_trim=<n>

<n> is the maximum number of non-ambiguous bases to try trimming from contig ends during healing.

--gap=<n>

<n> is the bases number of ambiguous 'n' bases to insert between contigs in the scaffold when a gap remains. Note that this base value may be adjusted by +1 or +2 to avoid introducing frame-shifts into long open reading frames that span a particular gap.

--verbose

Output extra diagnostic information to STDERR.

C.11 `tetracalc`

`tetracalc` is a high-performance tool for taxonomically clustering sequences using tetranucleotide (and trinucleotide) usage statistics. It works by calculating pairwise correlation coefficients of the "Z-statistics" (Teeling *et al.*, 2004) for all input sequences. Sequences are then binned using the pairwise nearest neighbor (Equitz, 1989) greedy clustering approach. As sequences are clustered together, the Z-statistics are intelligently recalculated for the merged bins [65] (Iverson & Riskin, 1993) so that as sequences are clustered they build statistical power. The pairwise clustering of sequence bins proceeds until no remaining pairwise combinations meet a cutoff threshold criteria. While it is possible to use fixed correlation thresholds, `tetra_calc` contains a large table of optimized sequence-length dependent thresholds, which have been carefully tuned against a large number of finished prokaryotic genomes taken from the RefSeq database, and this is the default (and recommended) way to use the tool in most cases. More information about SEAStAR scaffold clustering may also be found in the methods section of [64] (Iverson, *et al.*, 2012).

In most cases you will run the `tetracalc` tool indirectly from within `graph_ops` by using the `CLUST` command. However, it can be run independently.

C.11.1 Usage:

```
tetracalc [options] <infile1.fna> [<infile2.fna>...] > <outfile.json>
```

```
# Cluster sequences in input_seq.fna using default settings
tetracalc input_seqs.fna > output.json
```

`tetracalc` produces a simple custom JSON output file format:

```
{ "clusters" :     # An object with a single attribute
  [                                  # Which is a list
    ["NODE_123", "NODE_456", ... ],  # Of lists, of contig node ids
    ["NODE_987", "NODE_654", ... ],  # that clustered together
    ...
  ]
}
```

Note that `tetracalc` may also write warnings, errors and diagnostic information to STDERR, so it is important to keep STDOUT and STDERR separate when saving the output to a file, or the resulting saved JSON data may be corrupted by messages written to STDERR.

C.11.2 `tetracalc` optional parameters `[options]`:

```
[-h|--help] [--version] [--merge_tar=<n>]
[-m <n>|--min_len=<n>] [--num_threads=<n>]
```

`-h` / `--help`

Print a guide to valid command line parameters and their correct usage to the terminal, and then exit.

`--version`

Print the version number of the executing program and then exit.

`--merge_tar=<n>`

Target fraction of each genome merging to be merged into a single output bin. The valid of values for <n> is in the range 0.0-1.0 and the default is <n>= 0.95, which is a fairly conservative value, unlikely to lead to much sequence ending up in the wrong main genome bin. The value of `--merge_tar` represents the fraction of all sequence from a single genome that is likely to end up in the main (ie. largest by sequence length) bin corresponding to that genome in the output. Put more concretely, the default value means that ~95% (on average) of the sequence belonging to each distinct species-level genome in the input will cluster to-

gether in a single output bin, on average. Most of the remainder (%5 of the sequence in this case) will cluster into one or more smaller genome-specific bins; which can potentially be reconnected later using other information (e.g. pairing information). Lower values of <n> are more conservative because they are associated with higher correlation cutoff thresholds in the tuning tables. Conversely, values of <n> that are very close to 1.0 increase the likelihood of obtaining nearly complete genomes in each individual cluster bin, but at the expense of an increased chance that some sequences will be misclassified in the main genome bins in the output (as opposed to remaining unclustered, or in much smaller "birds-of-feather" bins of sequences from different genomes that are more similar to one another than they are to any single genome as a whole).

-m <n> / --min_len=<n>

Minimum sequence length for consideration. Default <n>= 5000 nucleotides. Shorter `--min_lengths` are possible, but very short sequences have poor statistical power for correlation using tetranucleotide statistics (the fall-back use of trinucleotides helps somewhat, but is less specific). There are 136 possible reverse-compliment unique tetranucleotide sequences (not 128, because 16 tetranucleotide sequences are palindromic); so for a pair of sequences to correlate well in their tetranucleotide usage, they must be long enough that the occurrence of relatively rare tetranucleotides can be accurately estimated; and that noise in the statistics due to undersampling is minimized.

C.11.3 Additional experimental / tuning [options]:

```
[--fixed] [-t <n>|--cor_thresh=<n>] [-s <n>|--sec_thresh=<n>]
[-r <n>|--tri_thresh=<n>] [-c <n>|--sec_len=<n>] [-f|--seq_files]
[-g|--greedy] [--chunk=<n>] [-z <filename>|--z_stats=<filename>]
[--verbose]
```

--fixed

Disable use of the `--merge_tar` tuned threshold tables, and instead use default fixed thresholds (perhaps modified by the parameters below).

`-t <n>` / **`--cor_thresh=<n>`**

Minimum tetra-nuc correlation (R value) for a link to be reported. Default <n>= 0.9

`-s <n>` / **`--sec_thresh=<n>`**

Secondary minimum tetra-nuc correlation for a sequence to be linked via a tri-nucleotide correlation. Default <n>= `--cor_thresh`

`-r <n>` / **`--tri_thresh=<n>`**

Minimum tri-nuc correlation for a sequence to be linked via a secondary correlation. Default <n>= `--cor_thresh`

`-c <n>` / **`--sec_len=<n>`**

Maximum sequence length for consideration of tri-nuc secondary correlation. Default <n>= `--min_len`

`-f` / **`--seq_files`**

Treat the sequences in each input file as a single scaffold.

`-g` / **`--greedy`**

Perform a greedy recalculation when merged seqs were the best correlation for other seqs.

`--num_threads=<n>`

Number of threads to use on multicore systems. Default <n>= NUM_PROCESSORS

`--chunk=<n>`

Break input sequences into chunks of N bases for analysis. Default: Do not chunk.

`-z <filename>` / **`--z_stats=<filename>`**

Output the Z-statistics for each input sequence/scaffold as a table to `<filename>`.

The format of the output table is tab separated values, with a header row:

```
Sequence AAAA AAAC AAAG AAAT ... TTTA TTTC TTTG TTTT
```

Where the column of each tetramer is one plus an 8-bit number composed of 2-bit encoded nucleotide values concatenated in sequence order:

```
A --> 00b   // 'b' denotes binary
C --> 01b
G --> 10b
T --> 11b

AAAA --> 00000000b + 1 = 1
AAAC --> 00000001b + 1 = 2
...
GCTA --> 10011100b + 1 = 157
...
TTTG --> 11111110b + 1 = 255
TTTT --> 11111111b + 1 = 256
```

Following the header, there is one row of tab separated values for each input sequence/scaffold, beginning with the sequence identifier and followed by the 256 Z-statistic values corresponding with the tetramers in the header row.

`--verbose`

Output extra diagnostic information to STDERR.

C.12 Miscellaneous scripts

There are a number of utility GAWK scripts that can be found in the source `scripts` directory and get copied to the build `bin` directory. Their names should give a reasonable summary of purpose and further descriptions can be found in header comments of the source.

C.13 FASTQ Format Reference

The SEAStAR tools use certain conventions for organizing read data in `.fastq` files. This section describes these conventions in sufficient detail to understand the operation of the tools described in this document.

`.fastq` files are named as: `<prefix>.[single|single1|single2|read1|read2].fastq`

`<prefix>` is a sample specific string used to identify a set of related read files.

`.read1.fastq` and `.read2.fastq` files contain the first and second reads in a pair respectively. Each read pair should be at the same position in both files.

`.single.fastq` files contain reads without a corresponding mate, either because the mate never existed (fragment run) or the mate was discarded during filtering.

`.single1.fastq` and `.single2.fastq` files contain reads without a corresponding mate and for which the original orientation of the read is known. This orientation information is useful for distinguishing sense from anti-sense transcription in strand-specific RNA-Seq data sets.

Each `.fastq` read ID should be formatted as `@<read_prefix>:<read_ID>`, where `<read_prefix>` is an identifier for this set of reads and `<read_ID>` is a read identifier which in combination with `<read_prefix>` forms a unique ID for this read. This combination only needs to be unique within reads of a single type, e.g. within all `.read1.fastq` reads.

`.fastq` files may be processed though a variety of quality control tools that modify (e.g. trim, error correct) and discard reads. Trimming can result in reads that vary in length, and since variable read length information is not always preserved through some downstream analyses (e.g. some alignment outputs), it is occasionally desirable to (optionally) preserve this information in the read header as well. The read length can be added to the beginning of read header as below, where `49|` indicates the actual number of nucleotides:

```
@49|lambda:1_68_381
```

```
CGAAACCGCCGGAACCAGCTTGTGCGGACGCGCCGGCCCCTGGACCGGA
+
9'4,)9,*%4/'#,32)*.1-1(5&3,%,$:*&#'8,+2'40$)05(,&
```

Quality scores are encoded as ASCII characters starting at a character value offset of 33, following the standard Sanger format for FASTQ files.

C.13.1 SOLiD

SOLiD reads and associated quality statistics from the instrument are packaged in FASTA-like format files called `.csfasta` and `.qual` files. Fragment (non-paired) libraries produce a single matched set of files named by convention as: `<prefix>_F3.csfasta` and `<prefix>_F3_QV.qual`

Paired libraries (either mate-paired or paired-end) additionally produce a second matched set of files named: `<prefix>_R3.csfasta` and `<prefix>_R3_QV.qual`

For some paired-end libraries, the `R3` in these (second mate) files may be replaced by `F5-BC` or `F5-P2`.

There is no guarantee that every read in an `F3` file has a corresponding mate in an `R3` file, or vice-versa. These unmated reads are called "singlets".

Most open source tools do not support these off-instrument SOLiD FASTA-like files, and so they must be converted to FASTQ format files for compatibility. This is complicated by the fact that SOLiD reads do not represent nucleotides, but are instead differential encoded colorspace reads, where each numbered color represents a coded di-nucleotide transition (e.g. `AA=0`, `AC=1`, etc). For colorspace reads to be translated into nucleotide space, an initial known base, the primer base, is required. The following example shows the colorspace sequence for one read prefixed with primer base `T`:

```
>1_68_381_F3
T0120001121122001102133232122012121122111132201122O
```

One simple approach to using these reads would be to simply convert them to nucleotide space `.fastq` files and from that point onwards ignore the colorspace source of the reads. Unfortunately, this does not work well because colorspace errors at any position in a read will effectively corrupt all of the resulting converted nucleotide sequence from the erroneous position(s) onwards. In addition, there are distinct potential advantages to processing (aligning and assembling) SOLiD reads in colorspace, delaying conversion to nucleotide space until later in the analysis pipeline when it can be done much more accurately (due to the error correction potential of the di-nucleotide encoding). For these reasons, the SEAStAR tools preserve the colorspace information in the FASTQ representation.

To accomplish this, we follow the convention used by MAQ, BWA and Velvet, mapping colorspace numbers to colorspace letters as `[0123.] → [ACGTN]`. Because the initial primer base and its following color value are not from the sampled DNA, and thus will not correspond to the actual sequence ~75% of the time, these two values need to be removed from the colorspace read data that is actually used for alignment or assembly. However, they need to be kept in some form so that the original nucleotide sequence of the read can be reconstructed at a later stage in the pipeline. To handle this, we have developed the following convention:

Original colorspace `.csfasta` read:

```
>1_68_381_F3
T0120001121122001102133232122012121122111132 20..2.0
```

Convert to colorspace letters:

```
>1_68_381_F3
TACGAAACCGCCGGAACCAGCTTGTGCGGACGCGCCGGCCCCTGGANNGNA
```

Convert to FASTQ format (with quality information added), relocating the primer base and following color `TA` to the header line (discarding the initial quality value).

```
@TA+lambda:1_68_381/2
CGAAACCGCCGGAACCAGCTTGTGCGGACGCGCCGGCCCCTGGACCGGA
+
9'4,)9,*%4/'#,32)*.1-1(5&3,%,$:*&#'8,+2'40$)05(,&
```

The read prefix `lambda` is added as an identifier for this set of reads, by default, from the `.csfasta` filename prefix. The read name suffix changes from the SOLiD specific `F3` to `/2`, designating it as the second mate or the only read in a fragment run. Similarly for mated reads, `R3` (or `F5-BC`/`F5-P2`) suffixes are changed to `/1`, designating them as the first mates.

If the resulting `.fastq` files are then be processed by `trimfastq` with `--add_len`, read lengths will be added to the read header after the primer base and first color. Below, the `49|` indicates the actual number of color encoded nucleotides:

```
@TA+49|lambda:1_68_381/2
CGAAACCGCCGGAACCAGCTTGTGCGGACGCGCCGGCCCCTGGACCGGA
+
9'4,)9,*%4/'#,32)*.1-1(5&3,%,$:*&#'8,+2'40$)05(,&
```

Reads conforming to these FASTQ conventions are used (and assumed) throughout the SEAStAR analysis pipeline, and any additional tools (custom scripts, etc) that modify FASTQ read data will need to preserve these conventions in their outputs.

C.14 JSON Sequence Graph File Format Reference

A JSON format "sequence graph" file is a data representation describing sequences as nodes (or vertices) in a mathematical graph. When pairing information is available, there are also edges. Edges are a summary of many connections between nodes formed by read pairs in which the two mates align with different sequences (connecting those sequence nodes in the graph).

The top level structure of a JSON sequence graph is:

```
{
  ### Attributes created by ref_select

  "SEASTAR_tool" : "",      # Path to ref_select executable used
  "SEASTAR_version" : "",   # Version of ref_select used
  "runtime_parameters" : {
    "parm" : "val", ...     # Object containing the ref_select
                            # parameters used
  },
  "nodes" : {
    <node_id> : { }, ...    # Object containing sequence node objects
  },
  "edges" : [          # List of pairing edges connecting sequence
                       # nodes
    { }, ...           # These edges are from mates spanning two
                       # sequences
  ],
  "internal_edges" : [ # List of pairing edges within nodes. These
                       # edges are from mates aligning to the
    { }, ...           # same sequence
  ],
  "shared_seq_edges" : [  # List of edges formed by multi-node
                          # mapping reads
    { }, ...            # These edges are from reads mapping to
                        # multiple
  ],                    # sequences, indicating duplication
  "rollup_stats" : {    # Optional, catalog per parent sequence
                        # statistics
        "seq_id" : {},  # A catalog file may link sequences together,
          ...           # into parent seqs, (e.g. scaffolds of a
                        # genome) Rollup calculates summary stats
  },                    # for the parent seq.
  "run_stats" : {
        "stat" : val, ... #  Run statistics generated by ref_select
  },

  ## In addition, graph_ops commands add other attributes to the
```

```
## sequence graph. Some or all of the attributes below may be absent
## depending on which commands have been run

"processing" : [        # List of graph_ops commands that have been
                        # run
  [ ], ...              # Each sublist contains details for one
                        # command
],
"removed_nodes: : {     # Same as "nodes" above, but currently
                        # unselected
  "node_id" : { }, ...  # Object containing sequence node objects
},
"removed_edges" : [     # Same as "edges" above, but currently
                        # unselected
  { }, ...              # These edges are from mates spanning two
                        # sequences
],
"digraph" : true,       # Indicates that the graph is now directed
"connected_comps" : [
  ["node1", ... ], ...  # List connected components. Sublists of
                        # node ids
],
"scaffolds" : {         # Object containing scaffolds, keyed by
                        # scaffold id
  "scaf0" : { }, ...    # Each scaffold is a sub object
},
"clusters" : [          # Same as the JSON output of tetracalc
  [ ], ...              # Each cluster is a list of scaffold ids
],
"merged_edges" : <bool> # When present (and true) indicates that the
                        # edges and mate-pairing statistics are
                        # originally from a separate (merged) JSON
                        # dataset. See the graph_ops LOAD command
                        # "edges" option.
  }
```

The nodes (and removed_nodes) object contains one attribute for each sequence node selected by ref_select. These attributes are keyed by <node_id>, and are themselves ob-

jects:

```
"nodes" : {
  <node_id> : {
    "bits" : <float>,       # Total bitscore of reads mapping to this
                            # sequence
    "rd_cnt" : <float>,     # Number of (perhaps fractional) reads
                            # mapping
    "int_cov" : <float>,    # Mapped reads per nucleotide (rd_cnt /
                            # seq_len)
    "rel_ab" : <float>,     # Relative abundance of this sequence,
                            # fraction of 1
    "cov" : <float>,        # Coverage, mean reads at each position in
                            # sequence
    "rd_len" : <float>,     # Mean read length of mapping reads
    "seq_len" : <int>,      # Reference sequence length in nucleotides
    "pct_uncov" : <float>,  # Percent of reference sequence uncovered
                            # by reads
    "mp_pairs" : <int>,     # Number of paired reads mapping within
                            # sequence
    "mp_sh" : <int>,        # Number of shared reads, mapping to other
                            # sequences
    "mp_fwd" : <int>,       # Number of reads with mates off 3' end of
                            # sequence
    "mp_bwd" : <int>,       # Number of reads with mates off 5' end of
                            # sequence
    "mp_ins_mean" : <float>,  # Mean insert length of pairs mapping
                              # within
    "mp_ins_stdev" : <float>, # Standard deviation of above insert
                              # lengths
    "name" : "",              # Name / unique node_id for this node
    "desc" : "",              # Text description of this sequence
    "adj_cov" : <float>,      # Adjusted coverage, omits near seq ends
    "adj_cov_stddev" : <float>, # Standard deviation of adj_cov
                                # within seq
    "adj_cov_min" : <float>,  # Minimum adjusted coverage
    "adj_cov_max" : <float>   # Maximum adjusted coverage
    "short_seq" : <bool>      # True if adj_* stats are omitted due to
                              # short seq
```

```
    "seq" : ""            # DNA sequence, optional, either
                          # reconstructed or provided
    "pct_gc" : <float>,   # GC composition as %, may be absent or NA
    "circular" : <bool>,  # True if this node represents circular DNA
                          # sequence
    "per_nt" : { ... }    # Sequence and per nucleotide stats are in
                          # this object
    "contig_problems" : [
      { }, ...            # List of potential problems found with
                          # this node
    ]
    "ref_str" : <bool>,   # Set by graph_ops. True if this sequence
                          # is on the "reference strand"
    "merged_stats" : { ... }  # Present when the "merged_edges"
                              # attribute is true.
  }, ...
}
```

The `per_nt` attribute of a node object is optional. If present it contains the per nucleotide information calculated for the node, including the DNA sequence itself, when present.

```
"per_nt" : {      # Per nucleotide statistics, all of these are
                  # optional.
  "mean_gc" : [
    <float>, ...  # Mean %GC content for nucleotides within a window
  ],
  "cov" : [
    <float>, ...  # Read coverage for each nucleotide position
  ],
  "phys_cov" : [
    <int>, ...    # Physical coverage of pairs spanning each position
  ],
  "mp_ins" : [
    <float>, ...  # Mean insert size for pairs spanning each position
  ],
}
```

The `contig_problems` attribute of a node object is optional, but if present it flags a num-

ber of different types of potential problems and pinpoints their location(s) within the sequence. These may also be viewed in `graph_ops` using the `PROBS` command.

```
"contig_problems" : [   # A node may have multiple problems,
                        # so it's a list of objects, one
  {                     # per problem region
    "start" : <int>,    # Staring coordinate of this problem
    "end" : <int>,      # Ending coordinate of this problem
    "type" :                          # What kind of problem?
       "Physical coverage break" |  # Contig should be split here...
       "Insert size anomaly" |      # Extra or missing sequence
                                    # detected
       "Collapsed duplication" |    # There are two/more repeated
                                    # sequences
       "Uncovered end" |            # Bad assembly / reconstruction
                                    # of end
       "Small insert"               # Sequence doesn't belong here...
  }, ...
]
```

The following attributes are present when the graph "merged_edges" attribute is true. These attributes preserve the node statistics of the graph that the edges in this graph came from.

```
"merged_stats" : {
  "bits" : <float>,     # Total bitscore of reads mapping to this
                        # sequence
  "rd_cnt" : <float>,   # Number of (perhaps fractional) reads
                        # mapping
  "int_cov" : <float>,  # Mapped reads per nucleotide
                        # (rd_cnt / seq_len)
  "rel_ab" : <float>,   # Relative abundance of this sequence,
                        # fraction of 1
  "cov" : <float>,      # Coverage, mean reads at each position in
                        # sequence
  "pct_uncov : <float>, # Percent of reference sequence uncovered by
                        # reads
```

```
  "rd_len" : <float>,   # Mean read length of mapping reads
  "adj_cov" : <float>,  # Adjusted coverage, omits near seq ends
  "adj_cov_stddev" : <float>,  # Standard deviation of adj_cov
                               # within seq
  "adj_cov_min" : <float>, # Minimum adjusted coverage
  "adj_cov_max" : <float>, # Maximum adjusted coverage
  "cov" : [                # This may be present when "merged_edges"
                           # is true
    <float>, ...  #  Read coverage for each nucleotide position from
  ]               #  an alternative mate-pairing alignment
}
```

The different `edges` lists (`edges`, `removed_edges`, `internal_edges`, `shared_seq_edges`) have similar syntax, but with different semantics:

```
# Note that internal and shared_seq edges will not have dir,
# org_dir or score attributes

"edges" : [           # Each edges list holds multiple edge objects
  {                   # each of which represents a relationship
                      # between:
    "n1" : "",        # A first node, referenced by name, and a
    "n2" : "",        # second node. This are the same for
                      # internal_edges
    "dir" : "",       # Direction of this edge: "FB", "FF", "BB" or
                      # "forward"
    "num" : <int>,    # Number of mapped read-pairs represented by
                      # this link
    "bits" : <float>, # Total bitscore of read-pairs participating in
                      # edge
    "p1" : <int>,     # Mean mapping coordinate in the first node
    "p2" : <int>,     # Mean mapping coordinate in the second node
    "score" : <float>,# Heuristic score for edge, calculated by
                      # graph_ops
    "org_dir" : ""    # Preserves original "dir" when edge becomes
                      # "forward"
  }, ...              # "dir" codes, where --> is:  5' --> 3'
]                     # "FB" :      --> edge -->
```

```
                    # "FF" :      --> edge <--
                    # "BB" :      <-- edge -->
                    # "forward" : directed (all "forward" edges are
                    # "FB" and on the "reference strand")
```

Rollup statistics are generated by `ref_select --rollup` parameter. They require use of a catalog file with `ref_select` that connects sub-sequences to their parents. A common use case is aligning reads to genome sequences, where some of the genomes may be unfinished and remain in numerous scaffolds. The rollup statistics summarize the stats by parent (genome) sequence, "rolled-up" from the underlying node sub-sequences.

```
"rollup_stats" : {
  "seq_id" : {
    "node_list" : [
      "node_id", ...      # List of node ids of the sub-seqs rolled
                          # into this
    ]
    "bits" : <float>,     # All of these statistics are the same as
                          # their named counterparts in the "node"
    "rd_count" : <float>, # object described above, except they
    "int_cov" : <float>,  # represent a proper amalgamation
    "rel_ab" : <float>,   # of the underlying sequence statistics, as
    "cov" : <float>,      # though all of the "parent" sequences were
    "rd_len" : <float>,   # simply a concatenation of their child
                          # subsequences. So for example, "rel_ab" is
    "seq_len" : <flaot>,  # relative abundance of this parent sequence
    "pct_gc" : <float>,   # compared to all other parent sequences.
    "pct_uncov" : <float>,
    "desc" : ""
  }, ....
}
```

Run statistics are `ref_select` overview statistics for the run that created this file

```
"run_stats" : {
   "num_selected_nodes" : <int>,               # Number of nodes in
```

```
                                                    # output
   "original_nodes" : <int>,                        # Number of original
                                                    # ref seqs
   "total_node_abundance" : <float>,                # Should be very close
                                                    # to 1.0!
   "mean_coverage" : <float>,                       # Optional: mean of all
                                                    # nodes
   "coverage_duplication_threshold" : <float>, # Thresh used for
                                                    # single genomes
   "reads_assigned_to_selected" : <int>,            # Reads mapping to
                                                    # output nodes
   "original_reads_aligned" : <int>,                # Reads mapping in
                                                    # alignment
   "second_chance_assigned_reads" : <int>           # Reads assigned to
                                                    # suboptimal
}                                                   # (but selected) node
```

The `processing` list is added by `graph_ops` to create a command history of changes made to this sequence graph.

```
"processing" : [
  [
    "script",   # Filename of the SCRIPT, '$' for cmd line or '>>'
                # for STDIN
    "version",  # SEAStAR version number of graph_ops used for this
                # command
    "command",  # Name of the graph_ops command executed

    { "parm" : val, ... }   # JSON parameters passed to the command
  ], ...
]
```

The `scaffolds` list is added by the `graph_ops SCAFF` command. It details the proper ordering of nodes in each scaffold, and links them back to connected components

```
"scaffolds" : {
  "scaffold_id" : {     # One object per scaffold
```

```
    "ccnum" : <int>,    # Reference to associated connected component
    "nodes" : [
      "node_id", ...    # This is an *ordered* list, 5' -> 3' of
                        # sequence
    ]                   # nodes, all on the same strand.
  }, ...
}
```

The `clusters` object is identical to the contents of the object output from the `tetracalc` tool:

```
"clusters" : [                      # A list...
  ["NODE_123", "NODE_456", ... ],   # of lists, of contig node ids
  ["NODE_987", "NODE_654", ... ],   # that were clustered together
  ...
]
```

VITA

Vaughn Iverson is a lifelong native of Washington State, USA.

Education:

- M.S., University of Washington, Seattle, Computer Science and Engineering.
- B.S. *cum laude*, Washington State University, Pullman, Computer Science.

Recent Honors:

- Mary Landsteiner Scholar, "For excellence in research at the intersection of interdisciplinary ocean science with advanced computing", School of Oceanography, University of Washington, 2012.

Selected Publications:

- V. Iverson, R. M. Morris, C. D. Frazar, C. T. Berthiaume, R. L. Morales, and E. V. Armbrust, Untangling genomes from metagenomes: revealing an uncultured class of marine Euryarchaeota, *Science*, Vol. 335 no. 6068 pp. 587-590
- T. Mock, M. P. Samanta, V. Iverson, C. Berthiaume, M. Robison, K. Holtermann, C. Durkin, S. S. BonDurant, K. Richmond, M. Rodesch, T. Kallas, E. L. Huttlin, F. Cerrina, M. R. Sussman, and E. V. Armbrust, Whole-genome expression profiling of the marine diatom Thalassiosira pseudonana identifies genes involved in silicon bioprocesses, *Proceedings of the National Academy of Sciences, USA* (PNAS), 105 (5): 1579-84
- V. Iverson, J. McVeigh and B. Reese, Real-Time H.264/AVC Codec on Intel Architectures, *Proceedings of International Conference on Image Processing* (ICIP) 2004, Volume II, pp. 757-760
- B. N. Schilit, A. LaMarca, G. Borriello, W. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong and V. Iverson, Ubiquitous Location-Aware Computing and the "Place Lab" Initiative, *The First ACM International Workshop on Wireless Mobile Applications and Services on WLAN* (WMASH)
- V. Iverson and E. A. Riskin, A Fast Method for Combining Palettes of Color Quantized Images, *Proceedings of International Conference on Acoustics, Speech and Signal Processing* (ICASSP) 1993, Volume V, pp. 317-320

Patents:

- 20 U.S. Patents awarded.