*

Genome Annotation

by

The SEED Team*

* See Appendix for Current Seed team

Table of Contents

Introduction **8**
- **The SEED Project** **8**
- **What is FIG?** **8**
  - The FIG Architecture: the SEED 9

Components of the SEED Annotation System **11**
- **The SEED, the A-SEED, the P-SEED, and the PubSEED** **11**
  - SEED 11
  - Annotator's SEED (A-SEED) 11
  - PATRIC SEED (P-SEED) 11
  - Public SEED (PubSEED) 11
- **What SEED Do I Use?** **11**
  - Ownership of Genomes 12
  - Setting Up an Annotation Group 15
  - Access of Data Via the Servers 15
  - Maintenance of Subsystems 16
- **Subsystems** **17**
- **FIGfams** **17**
- **RAST** **18**
- **Metabolic Modeling** **19**
- **Model Tutorials** **19**
- **Metagenomics** **20**

Annotating your Genome **21**
- **Basic Steps in Annotating a Prokaryotic Genome** **21**
  - Running Your Genome Through RAST 21
  - "Walking" Your Genome 21
  - Building a Metabolic Reconstruction 22
- **Summary** **23**
- **Annotating a Genome Using RAST** **23**
  - Running Your Genome Through RAST 24

Walking your Genome Using RAST 24

Exporting Your Genome from RAST 29

Annotating a Genome with myRAST 29

Running a Genome Through myRAST 30

Walking your Genome Using myRAST 32

Exporting Your Genome from myRAST 33

Metabolic Modeling 35

Modeling Overview 35

Accessing The SEED Database 37

The Entity-Relationship Model 37

Summary of Individual Servers 38

The Sapling Server 38

The Annotation Support Server 38

The RAST server 38

The Model Server 39

Getting Started 39

Getting started with Command Line "svr" scripts 39

Getting started writing Perl scripts to access the servers 41

Using the Command Line Scripts 43

A (Very) Minimal Introduction to Some Basic Command-Line Tools 43

Command Line Services 47

Find all features for a genome. 48

Find Gene Function 48

Find Gene Aliases 49

Find Neighbors 50

More Examples 51

The RAST Batch Interface 51

Advanced Programming with the Servers 54

Getting a list of Genomes and Their Taxonomies 54

Listing All Genomes 54

Taxonomy 55

Retrieving Features and Functions for a Genome 56
Conversion of Gene and Protein ID's 58
Example 1 Discussion 59

Output Table 60

Metabolic Reconstructions Provided for Complete Prokaryotic Genomes 60
Example 2 Discussion 60

Example 2 Input File (Truncated) 61

Example 2 Output Table (Truncated) 62

Locating Functionally Coupled PEGs 63

Using the Servers for Genome Annotation 66
Annotating a genome using the SEED servers (via the command line or Perl code) 66
Calling RNA-Encoding Genes Using a Command Line Script 66

Accessing Annotation Services from a Perl Program 68

Extending the ends of the contigs. 70
Abstract 70

Sequence Quality 70

End locations 71

Example 1: 79

Example 2. 81

Example 3 88

Example 4. 90

Concluding thoughts 92

Conjectures 93
Formulating Conjectures: Using the Browser and Atomic Regulons 93
Formulating Conjectures: Using the Browser and Atomic Regulons - Part 2 94
1. fig|300852.3.peg.2216: an Example Relating to CRISPRs 94

2. fig|224911.1.peg.1749 in Bradyrhizobium japonicum USDA 110 95

3. fig|224911.1.peg.1443 in Bradyrhizobium japonicum USDA 110 95

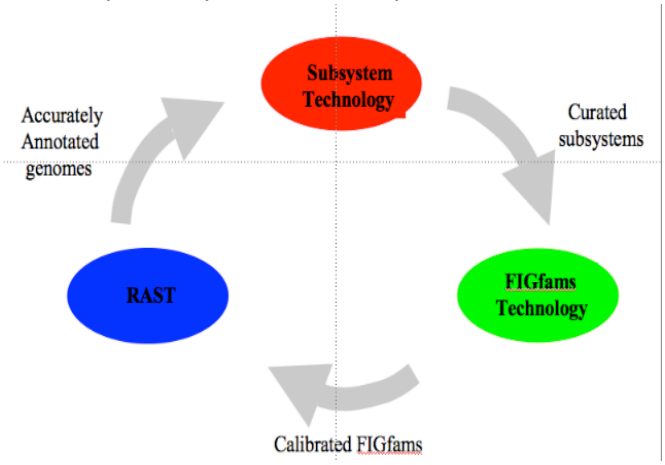
4. fig|211586.9.peg.2693 in Shewanella oneidensis MR-1 95
5. fig|211586.9.peg.2892 in Shewanella oneidensis MR-1 96
6. fig|211586.9.peg.4166 in Shewanella oneidensis MR-1 96
7. fig|100226.1.peg.4182 in Streptomyces coelicolor A3(2) 96
Differential Expression Analysis tool 97
Select a genome 97
Select expression samples 97
Differential Expression Results 98
Excersise: 98
Some Notes on How to Look for Co-expressed Genes Using the Expression Data ..100
A Case Study in Use of the "Server Scripts" 103
Searching for UDP-2,3-diacylglucosamine hydrolase (EC 3.6.1.-) in *Agrobacterium tumefaciens str. C58* 105
"Lipopolysaccharide core biosynthesis protein RfaZ" in *Pseudomonas aeruginosa PAO1* 107
An Exercise in Tools for Formulating Conjectures 115
Introduction 115
Finding the Neighborhood of a Reaction/Role 117
Inverting the Approach 118
Analysis of Metagenomics using the SEED 120
Getting Summaries of Functional Content and OTUs for an Metagenomic Sample 120
An Etude Relating to a Metagenomics Sample 121
Appendix 127
The SEED Team 127
FIG 127
ANL/UofC 127
UIUC 127
Hope College 127
SDSU 127
SVR Routines 128
Annotation and Assertion Data Methods 128

Annotation Support 128
DNA and Protein Sequence Methods 128
Expression Data Methods 128
Feature (Gene) Data Methods 129
FIGfam Data Methods 129
Functional Coupling Data Methods 129
Genome Data Methods 129
Subsystem Data Methods 130
Alignment/Tree Methods 130
Chemistry Methods 131
Gap Filling Support (finding missing genes) 131
Utility Methods 132
Annotated list of Getting Started, Tutorial and Coding Examples 132
Publications describing SEED, RAST, and NMPDR tools 135
SEED and RAST 135
NMPDR 136
Publications using SEED, RAST, or NMPDR tools 137
2011 137
2010 137
2009 141
2008 144
2007 148
2006 150
2005 152
Index 153

Introduction

The SEED Project

With the growing number of available sequenced genomes, the need for an environment to support effective comparative analysis increases. The original SEED Project was started in 2003 by the Fellowship for Interpretation of Genomes (FIG) as a largely unfunded open source effort. Argonne National Laboratory and the University of Chicago joined the project, and now much of the activity occurs at those two institutions (as well as the University of Illinois at Urbana-Champaign, Hope college, San Diego State University, the Burnham Institute and a number of other institutions). The cooperative effort focuses on the development of the comparative genomics environment called the SEED and, more importantly, on the development of curated genomic data. Curation of genomic data (annotation) is done via the curation of subsystems by an expert annotator across many genomes, not on a gene-by-gene basis. This is also detailed in our manifesto. From the curated subsystems we extract a set of freely available protein families (FIGfams). These FIGfams form the core component of our RAST (Rapid Annotation using Subsytems Technology.) automated annotation technology. The RAST technology provides automatic Seed-Quality annotations for more or less complete bacterial and archaeal genomes.



What is FIG?

FIG is a nonprofit organization devoted to providing support for those analyzing genomes. Sequencing of genomes is laying the foundation for advances in science that will dramatically reshape our society. These advances will initially occur in medicine, agriculture, and chemical production, but in the long term the impact will be pervasive. The computer revolution started by impacting payrolls, but eventually allowed man to travel to the moon. Similarly, the biological revolution is beginning by reshaping the life sciences, but this will surely not be the whole story or even the most significant outcome.

The interpretation of genomes will constitute the most exciting and most significant science of the century. By rapidly advancing our understanding of life, how it arose, and how it continues to change, we will acquire the tools that will allow us to better understand and improve our existence. Understanding will begin with relatively simple forms of life -- unicellular organisms. While the central mechanisms of life are shared by both these organisms and the most complex animals and plants, they also contain a remarkable diversity. They have an immense amount to teach us about life itself, and we will need to master these lessons before full understanding of complex genomes will be achievable.

The Fellowship for Interpretation of Genomes will focus on organizing the data needed to support interpretation of genomes, providing the infrastructure needed by the world community in its efforts to achieve understanding. In addition, we will ourselves pick specific, critical problems and attempt to actively participate in the unraveling of the secrets within these amazing entities. It is only by merging the work of building infrastructure with the applications that use it that we will more deeply understand what is needed at each step.

FIG was started in May 2003. The founders were Michael Fonstein, Yakov Kogan, Andrei Osterman, Ross Overbeek, and Veronika Vonstein. An early position paper began with the following comments:

The FIG Architecture: the SEED

We begin with the "seed" of FIG. The SEED contains the essential, basic elements that are needed to sustain a scalable integration of thousands of genomes. The later parts of this document will attempt to offer precise notions of what makes up the seed of FIG. I will cover the basic types of objects, make comments on what extensions will be needed to support hundreds of thousands of genomes, and offer an implementation plan.

However, before we go into such detail, some broad notions should be discussed. The idea of integrating hundreds of thousands of genomes needs some clarification. Indeed, what is meant by integrating a bunch of genomes, no matter what the number. In my mind, the notion of integration is essentially "maintenance of notions of neighborhood, allowing forms of access that can be used to easily explore connections and comparisons between data from numerous genomes". This may be viewed as a complicated way to say "a framework to support comparative analysis".

To be more precise: Genes from single genomes are often "functionally related" in that they participate in implementing a single pathway or subsystem. For any single gene, the "functional neighborhood" of that gene is the set of genes that are functionally related to the gene. To support access relating to this notion of neighborhood requires an encoding of the cellular machinery (e.g., pathways). Genes that occur close to each other on a chromosome may be thought of as "positionally related". The set of genes that are positionally related to a given gene amounts to the "positional neighborhood" of the gene. One of the huge payouts of integrations to data has been based on a correlation between the neighborhoods imposed by "functionally

related" and "positionally related" in the case of prokaryotic genomes. Genes from one or more genomes that share a common ancestor are called "homologous". Homology induces yet another notion of neighborhood. One can build more restricted neighborhoods upon this basic concept. Thus, we tend to think of a protein family as a set of homologous genes that have a common function (an imprecise notion, we grant). Maintenance of protein families will, of course, be an absolutely essential part of effectively integrating many thousands of genomes. Sets of very closely related genomes may be viewed as a neighborhood (i.e., the neighborhood of a genome becomes a set of closely related genomes). One can layer a notion of "variation", including SNPs, on the notion of closely related genomes, and then whole frameworks for exploring minor variations become possible. The power in an integration arises from mixing the different notions of neighborhood. The tools for supporting effective use of a variety of comparative notions constitute the computational framework for comparative analysis, which is often abbreviated to the notion of "integration".

FIG offers the key services required to architect and implement a comparative framework for interpreting genomes.

The Fellowship for Interpretation of Genomes is a 501 (c) (3) organization.

See the appendix for a complete list of the current members of the SEED team.

Components of the SEED Annotation System

The SEED, the A-SEED, the P-SEED, and the PubSEED

SEED

A SEED is an integration of genomic, expression, regulatory and modeling data constructed using the tools provided by the SEED Project. Specific instantiations of the SEED are used to support distinct projects or goals. For example, we maintain three distinct SEEDs at Argonne National Lab, which we describe below. Other groups at universities have built and maintained their own SEEDs, but this requires active participation in the SEED Project to keep track of all the needed components.

Annotator's SEED (A-SEED)

The A-SEED is a copy of the SEED used cooperatively by an international group of researchers to annotate genomic data by constructing subsystems. This copy of the SEED has been the core of the SEED annotation effort. It contains a representative set of about 1000 genomes.

PATRIC SEED (P-SEED)

The P-SEED contains all of the complete prokaryotic genomes deposited to Genbank. It is used to support the PATRIC database, which is supported by NIH to facilitate research on microbial pathogens.

Public SEED (PubSEED)

The PubSEED is a SEED that is open to anyone. Anyone who wishes to annotate or build subsystems will need to become a registered RAST user (whether or not they ever intend to use RAST; see the video tutorial on how to register: http://blog.theseed.org/servers/2010/08/video-tutorial-creating-a-rast-account.html).

For a detailed and somewhat technical description of how the annotations and FIGfams are updated, see The Update Protocol for Maintenance of Annotations.

What SEED Do I Use?

March 17, 2011

Ross Overbeek

A number of distinct SEEDs have emerged over the years. Almost all users will find they wish to use just the following two:

1. The **PubSEED** will rapidly become the central SEED for use by the research community. It will support access to the largest collection of genomes. The constant influx of new genomes will be to the PubSEED first. The PubSEED will support the ability for registered users to make annotations and subsystems (unfortunately, that implies that they will be able to overwrite the work of others, too). We will support the ability for registered users to install genomes from RAST directly into the PubSEED. Access and update capabilities, by genome, will require establishing a notion of *ownership of genomes.* Users will be allowed to *copy a genome* creating a version with ownership rights that they control. We will architect a few basic rules, and then we will do our best to develop tools that support reconciliation of conflicts, backup and recovery to specific points in time, and so forth. This SEED will be the center of much of our work.

2. The **UC-SEED** (i.e., the University of Chicago SEED) will be rebuilt periodically as a copy of the PubSEED. It is a place where classes can be held, students can do annotations and build subsystems, and so forth. Subsystems built in the UC-SEED can be *exported to the Clearinghouse*, and then *imported to the PubSEED*. This procedure allows users to save work and make it available, if they wish. However, the whole system is rebuilt and the existing contents destroyed on a periodic basis (usually between semesters, and after several weeks in which there will be a posted notice on the front page).

Ownership of Genomes

I am now discussing a basic position relating to genomes that is not yet fully implemented. I believe that it will move to the position I describe within 4-6 months.

There will soon be hundreds of thousands of genomes. Many of these genomes will be either identical or almost identical. In some cases the genomic sequence data will be identical, but distinct user groups will insist on the ability to annotate isolated copies that are protected from unauthorized updates. Determination of a protocol that effectively supports both sharing and

isolated annotation will require support for effectively managing privileges and interactions in a way that minimally constrains experts attempting to contribute.

It will rapidly become critical that we be able to talk about genomes, contigs, genes, and proteins and to easily detect whether two references are to the "same" entity. As we move into a world with hundreds of thousands of genomes, some with identical sequence, and others with sequences that differ by only a few characters, it will become critical that we support basic **ID Correspondence Services** in a consistent manner.

We suggest employing the following set of definitions for what it means to be "the same" for **genomes, contigs, genes,** and **proteins.**

1. Two sequences are the same if the MD5 functions of the uppercase versions of the sequences are identical.

2. Two contigs are considered the same if their DNA sequences are the same.

3. Two genomes are considered the same iff
    a. They have the same number of contigs.
    b. The MD5 function of the sorted and concatenated contig MD5s match. We call the MD5 function of the sorted and concatenated contigs the **MD5 of the genome.**

4. Two genes are considered identical if they are in genomes that are the same, and
    a. They occur in contigs that are the same,
    b. They have identical start and stop positions in the two contigs.

5. Two proteins are the same if their sequences are the same (note that this is not a notion that is equivalent to saying that they are the gene products of two genes that are the same).

We will support the ability to rapidly determine which genomes, genes, and proteins are identical. Further, we will support the capability of users defining sets of representative genomes and limiting displays to any selected set.

We are architecting the SEED environment as a framework that will be able to effectively integrate initially thousands, and within a short period millions, of distinct genomes. Genomes will enter the collection from a growing number of sources. *Registerying a Genome* will amount to claiming unique IDs for the genome and the features that occur within the genome. This will inevitably lead to multiple registrations for identical genomes. Further, while we will not support alteration of the sequence of a genome (i.e., such a change would lead to the creation and registration of a new genome), we will support addition and deletion of features on a genome. A deletion will lead to recording a change in status (retaining a complete record of the deleted feature indefinitely). The addition of a feature would require the acquisition on a new ID. Changing the start location of a gene would cause deletion of the existing feature and addition of a new feature, which would inherit the appropriate attributes from the deleted feature.

The SEED environment will support the maintenance of genomes and features via a set of services that will include:

1. acquire_a_genome_ID returns a genome ID to a registered user
2. acquire_a_feature_ID(Genome,Contig,Start,Stop) returns a feature ID
3. delete_a_feature(ID) requires a update privileges
4. reactivate a deleted genome(ID) requires update privileges

Registered users will be able to make any of these operations against genomes for which they have the required privileges. Users owning genomes will have the ability to restrict access to a specified set of users. That is, we will support *private genomes* that are not seen by everyone, and we will support the ability of owners to change the status of a genome (from *private* to *public* and vice versa).

Perhaps a short summary of the decision procedures on access/update rights would be as follows:

1. We have **registered users.** Users are either **superusers** or **normal users.**
2. We have genomes. Genomes are either **private** or **public.**
3. Anyone attempting to access a genome or a feature of a genome will be given access if and only if
    a. the genome is public, or
    b. the user is a superuser, or
    c. the user either owns the genome or has been granted access to the genome.

4. Anyone attempting to update a genome (which includes annotating features, deleting features, and adding features) will be allowed to make the update iff and only if
    a. the genome is public, or
    b. the user is a superuser, or
    c. the user either owns the genome or has been granted write privileges to the genome.

Setting Up an Annotation Group

If a group wishes to use the SEED Environment as a resource for supporting annotation and analysis of their genome, they would begin by registering each member of the group, and then establishing a group containing those members.

They would select the genomes they wish to annotate (probably by importing a newly-annotated genome from RAST into the PubSEED). They would decide whether access and update privileges should be restricted to the group or not.

Then, they would use the framework we currently use to support our annotators to examine and edit annotations, construct metabolic models, or whatever. The set of genomes that would be simultaneously be edited could all be public or all be private. If private, they would be imported from RAST or as copies of existing genomes.

Access of Data Via the Servers

Most users of the SEED will use a web browser. However, a growing body of users will also start using our **SEED Servers**, which support a well-defined API to access and update data from a SEED. We will run servers for the PubSEED. See

http://servers.nmpdr.org/servers

for a discussion of the servers, the APIs used to access them, and the command-line services supported via the servers. We believe that research groups may wish to use or help extend this growing confederation of servers.

Maintenance of Subsystems

The PubSEED, and UC-SEED both support development of subsystems. From any of these platforms, subsystems can be exported to the **Clearinghouse**, and they can be imported into any other SEED (if you have the appropriate privileges). We anticipate that students in classes would use the UC-SEED to avoid destroying the work of others. Users wishing to make a more permanent contribution would use the PubSEED.

We will try to install a few basic rules to prevent bloodshed in instances in which incompatible annotations must be reconciled. They would be something like

1. You may overwrite any annotation that is not in a subsystem or is a duplicate in a subsystem (i.e., a case in which two genes currently have the same assigned functional role).

2. Before overwriting a function in which a gene plays a unique role in someone else's subsystem, email them and ask for permission. If they do not respond with a few days, proceed.

As the number of genomes grows rapidly, we believe that fewer and fewer annotators will actually construct and maintain comprehensive subsystems. Rather, there will be a growing number of subsystems that contain only a subset of the actual genomes that have the machinery. To handle this situation, we will periodically produce estimates, for each genome, of the subsystems that should contain the genome. These will not impact any of the subsystems, but will allow users to have a reasonable estimate of the molecular machinery that can be identified. This mimics what is now done in RAST, where a new genome contains estimates of which genes go into which subsystems, but these estimates do not actually impact the subsystems themselves.

The real point is that subsystems will no longer be thought of as comprehensive. Up to this point the goal was to provide the tools needed to support manual curation of subsystems that were to contain as many genomes as possible from the existing collection. The goal will shift. We will think of subsystems as containing a diverse collection of instances needed to support accurate projection over the entire collection. The PubSEED will be used to house as complete a collection as possible, to support experimentation, and will inevitably lead to conflicts (that, hopefully, enrich the overall collection and get resolved peaceably).

Subsystems

The use of subsystems as a key technology for annotation of genomes was introduced in The Subsystems Approach to Genome Annotation and its Use in the Project to Annotate 1000 Genomes. We recommend reading this paper for a detailed discussion.

A subsystem is a set of functional roles that together implement a specific biological process or structural complex. A subsystem may be thought of as generalization of the term pathway. Thus, just as glycolysis is composed of a set of functional roles (glucokinase, glucose-6-phosphate isomerase and phosphofuctokinase, etc.) a complex like the ribosome or a transport system can be viewed as a collection of functional roles. In practice, we put no restriction on how curators select the set of functional roles they wish to group into a subsystem, and we find subsystems being created to represent the set of functional roles that make up pathogenicity islands, prophages, transport cassettes and complexes (although many of the existing subsystems do correspond to metabolic pathways). The concept of populated subsystem is an extension of the basic notion of subsystems - it amounts to a subsystem along with a spreadsheet depicting the exact genes that implement the functional roles of the subsystem in specific genomes. The populated subsystem specifies which organisms include operational variants of the subsystem and which genes in those organisms implement the functional roles that make up the subsystem. Each column in the spreadsheet corresponds to a functional role from the subsystem, each row represents a genome, and each cell identifies the genes within the genome that encode proteins which implement the specific functional role within the designated genome.

At this point (August, 2010), over 1200 subsystems have been constructed, containing over 11,000 distinct functional roles and 1,400,000 PEGs (genes). Many of these subsystems have been "experimental" in the sense that they were constructed to support specific hypotheses and then not maintained. As many as a third of the collection fall into this category.

See The Project to Annotate 1000 Genomes for our manifesto written in 2004 describing a basic strategy for creating a framework to support high-throughput annotation. For a brief presentation on this subject, see http://blog.theseed.org/servers/documents/subsys.pdf

FIGfams

Each FIGfam is a set of proteins that are believed to be isofunctional homologs. That is, they all are believed to implement the same function, and they are believed to derive from a common ancestor because they appear to be similar. Given two members of a FIGfam, it should be the case that they can be globally aligned.

FIGfams are generated in two ways:

1. They are derived from subsystems (the set of PEGs in a column that are globally similar becomes a FIGfam).

2. We have tools that align closely-related genomes, and genes that appear to "clearly correspond to one another" are placed in the same FIGfam.

Note that there is no manual curation of FIGfams. They are automatically derived. The manual annnotation occurs within the subsystems. If errors are detected within a FIGfam, the correction is made by fixing a subsystem or creating a new subsystem -- causing the derivation process to produce improved FIGfams.

At this point, there are multiple FIGfam collections. The largest contain over 130,000 sets of proteins (of which about 50% of the sets contain only two sequences).

For a brief presentation on FIGfams, see this PDF.

RAST

The RAST server was brought up in 2007 and we published a description of the technology in 2008 The RAST Server: Rapid Annotations using Subsystems Technology.

The basic server was designed to support rapid annotation of prokaryotic genomes using subsystems technology. We believe that the system is both unusually fast and unusually accurate.

RAST bases its attempts to achieve accuracy, consistency, and completeness on the use of a growing library of subsystems that are manually curated and on protein families largely derived from the subsystems (FIGfams).

The RAST server automatically produces two classes of asserted gene functions: subsystem-based assertions are based on recognition of functional variants of subsystems, while nonsubsystem-based assertions are filled in using more common approaches based on integration of evidence from a number of tools. The fact that RAST distinguishes these two classes of annotation and uses the relatively reliable subsystem-based assertions as the basis for

a detailed metabolic reconstruction makes the RAST annotations an exceptionally good starting point for a more comprehensive annotation effort.

Besides producing initial assignments of gene function and a metabolic reconstruction, the RAST server provides an environment for browsing the annotated genome and comparing it to the hundreds of genomes maintained within the SEED integration. The genome viewer included in RAST supports detailed comparison against existing genomes, determination of genes that the genome has in common with specific sets of genomes (or, genes that distinguish the genome from those in a set of existing genomes), the ability to display genomic context around specific genes, and the ability to download relevant information and annotations as desired.

To date, users have submitted over 14,000 jobs to the RAST server. We are planning enhancements to support processing phages, plasmids, and short fragments of DNA. We are also developing a desktop version of RAST, called myRAST, which will run on users' laptops (we will be targeting Macs and Windows machines initially).

Metabolic Modeling

When we say that we now support generation, maintenance, and use of "metabolic models", what do we mean? There are a number of possible meanings of such a term, and many of them are used in different contexts.

For our purposes a metabolic model is three things:

1. the biomass reaction, which is a list of small compounds, co-factors, nucleotides, amino acids, and cell wall components needed to support growth. We think of this as the list of "required parts".
2. a list of the compounds that can be transported into and out of the cell
3. the reaction network that the cell uses to maintain its existence. This reaction network is encoded as a stoichiometric matrix.

We have defined a precise encoding of models, so we can import and export them, as well as updating them to reflect constantly improving estimates of the roles of specific genes and knowledge of phenotype.

Model Tutorials

Annotations to Reactions

Editing Your Model
Generating Predictions on Your Model
Model Phenotypes
ModelView Presentation PDFs
Reactions to Initial Model
Viewing your Initial Model

Metagenomics

Increasingly, we are receiving queries from users with metagenomic samples asking if they can use the SEED to examine their samples. Using our servers, we have developed methods for obtaining summaries of functional content and OTUs for a metagenomic sample. These methods for studying metagenomic samples will be described in detail in later sections.

Annotating your Genome

Basic Steps in Annotating a Prokaryotic Genome

As it becomes possible to quickly and cheaply acquire the genomes of organisms, the need to produce accurate annotations quickly has become more pressing. This short tutorial is designed to enable a user to produce relatively accurate annotations quite quickly (under a week for most prokaryotic genomes). The steps we will describe are as follows:

1. First, submit the contigs representing the sequence of the organism to the RAST server (or any similar server), which produces an initial annotation.

2. Then, we advocate "walking your genome" rapidly to gain a sense of how closely it matches existing (previously sequenced and annotated) genomes, to delete clearly miscalled genes, and to gain an understanding of the number of potential problems (e.g., frameshifts) that exist. We suggest correcting any clearly improvable functions that may have been assigned incorrectly in step 1 as you walk through the genome.

3. Automatically place the genes into subsystems, giving an overview of the cellular machinery that has been successfully identified.

These three steps are just the start of extracting information from a new genome, but they do offer a technology that will give you a reasonably annotated genome that can be used effectively by the research community.

Running Your Genome Through RAST

The first step involves acquiring an initial annotation. We suggest that you use RAST or our MacApp for doing so, but there are other services and approaches to getting an initial annotation.

Go here to see a tutorial on how to get a RAST account and submit a genome for annotation.

"Walking" Your Genome

However you decide to manually annotate your genome, we suggest using an environment that supports efficiently "walking through the genome" comparing regions against those in previously sequenced and annotated genomes. This can be done quite rapidly if you use a suitable framework. Here we are talking about visually inspecting all of the genes in about 1 to 3 workdays. This can be somewhat tedious, but what emerges is a reasonably annotated genome for which you have a pretty good overview of what is there.

Building a Metabolic Reconstruction

It is useful to group the recognized genes into the recognized pathways, complexes, and nonmetabolic molecular machines. Here is how we view this process:

1. Our annotation team has constructed sets of functional roles that are annotated simultaneously because the functional roles are related. The roles may be distinct subunits of a complex (e.g., the subunits of the ATP synthase or the ribosomal proteins), a set of functional roles that constitute a pathway (e.g., Histidine Degradation) or the genes may make up a nonmetabolic molecular machine (e.g., a repair machine, a transport cassette, or a 2-component regulatory system). We call each of these sets of roles a "subsystem". Our annotators have carefully assembled the functional roles that make up a subsystem and for each one constructed a spreadsheet in which each row is a genome and each column is a distinct functional role. The cells of the spreadsheet contain the genes from the specific genome that implement the specific functional role. For example (SEE POWERPOINT PICTURES OF HISTIDINE DEGRADATION).

2. We automatically, using the examples contained in the manually curated set of subsystems, try to locate the appropriate genes within the newly sequenced genome and identify a new instance (i.e., a new row in the spreadsheet) of the subsystem. When we can identify all of the genes needed to implement an operational version of the subsystem, it substantially increases the confidence we have in the assigned functions, and it forms a critical piece of information needed to support the generation of metabolic models.

3. Where we recognize a portion of a subsystem, we may have failed to accurately identify some genes, we may have mis-annotated genes, or we may have a new variant of the subsystem (e.g., a new variant of a common pathway),

4. We consider a metabolic reconstruction to simply be the set of recognized, operational instances of our subsystem collection. This is distinct from an actual initial estimate of the metabolic network (which we provide, as well). The metabolic reconstruction includes information about the nonmetabolic machinery supported by the genome. We are not completely happy with the term "metabolic reconstruction", but that is the term that has stuck and the one in common usage within our group.

Summary

The 3-step process we outline for acquiring reasonably good annotations and an initial annotation for a prokaryotic genome works well for genomes that are "close" to well-annotated existing genomes. For truly divergent genomes, it is a good starting point, but much more effort is required to achieve what one might think of as an "acceptable annotation". The virtue of our approach is that, in most cases, you can acquire a usable annotation in 1-3 days. We have invited groups that have spent man-years annotating specific genomes, and for the most part our annotations were very close to the carefully done manual efforts.

Annotating a Genome Using RAST

With RAST, it is now possible to get a fairly accurate annotation of a prokaryotic genome in about a day. We believe that the result is often very close to what most annotation groups can produce spending months or even man-years. This short tutorial describes our recommended approach to producing a rapid, quite-accurate annotation within about a day (sometimes less for short genomes, and often more for lare or diverged genomes).

The approach that we advocate is especially suited to annotating a genome that is quite phylogenetically close to an existing (presumably, well-annotated) genome or set of genomes. In particular, it works well for newly-sequenced pathogen genomes that are close to large groups of already sequenced genomes.

The proposed approach is as follows:

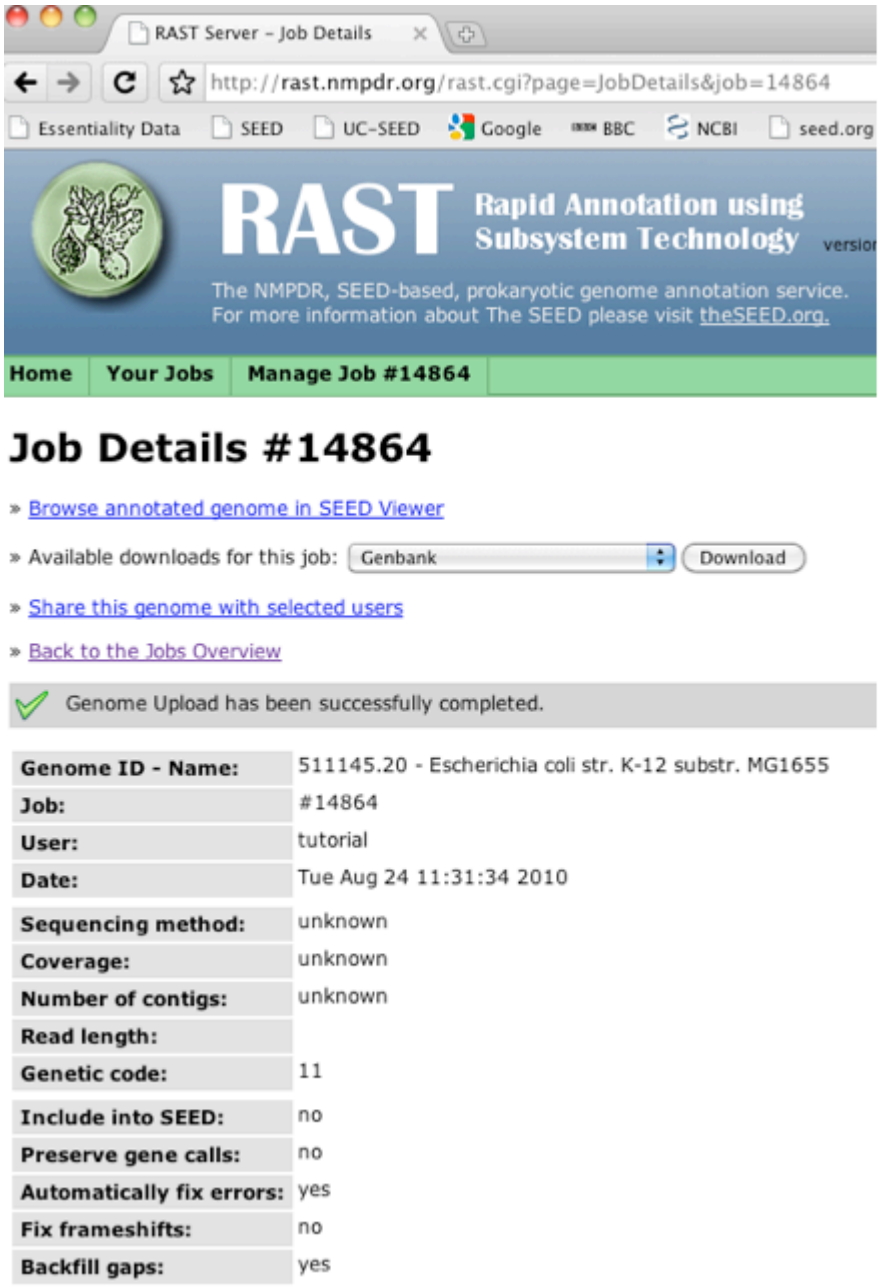
1. Run your genome through RAST. This produces an initial annotation. There will probably be errors in gene calls, as well as errors in the assigned functions. Those get cleaned up in the next step.
2. Once you have produced an initial annotation, you can "walk the genome" looking for genes that need to be deleted, inserted, or just re-annotated.
3. Once you have made a quick pass through the genome, we suggest that you export the genome. You will probably wish to do this twice -- once to produce a Genbank formatted version (which can be used by many tools) and a second as a set of tab-separated files suitable for perusing in a tool like Excel.

Running Your Genome Through RAST

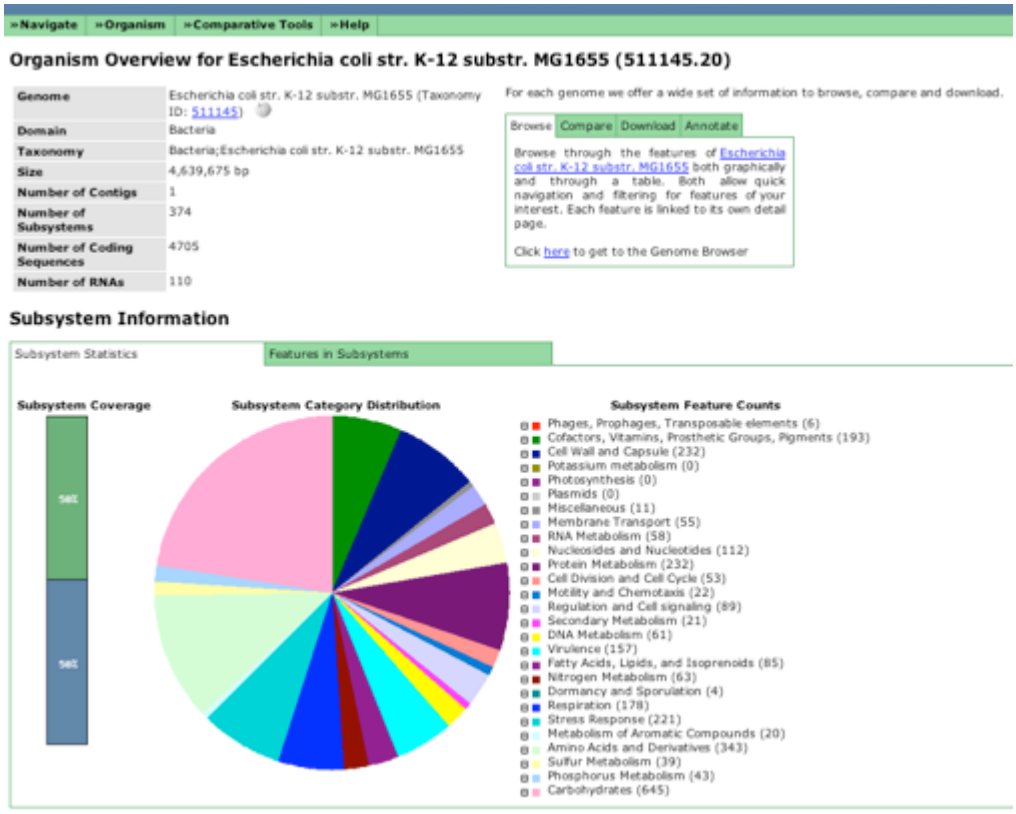
For detailed instructions on how to get a RAST account, how to submit a genome for annotation and so forth, go to the writeup on using RAST in the SEED Servers Blog. The writeups there should help you get started. If you need help, you can email us as RAST@mcs.anl.gov, but please realize that we are processing jobs for over 3000 users at this point.

Walking your Genome Using RAST

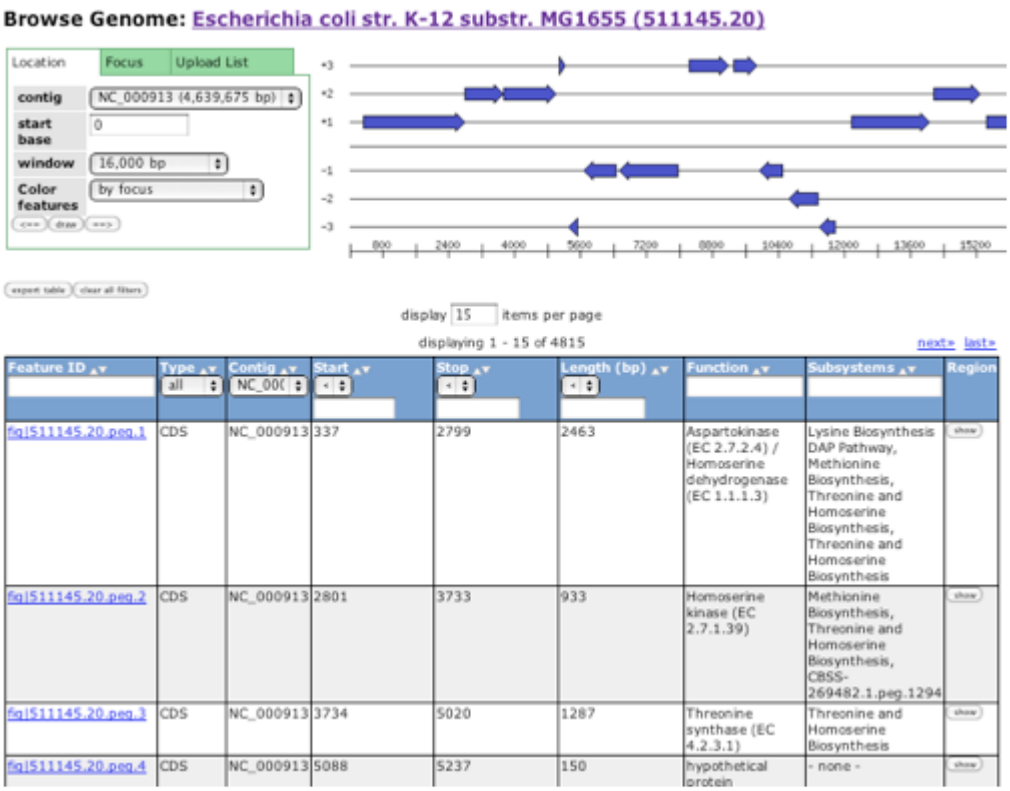
To begin looking at your annotated genome, you start at the "Job Details" page:



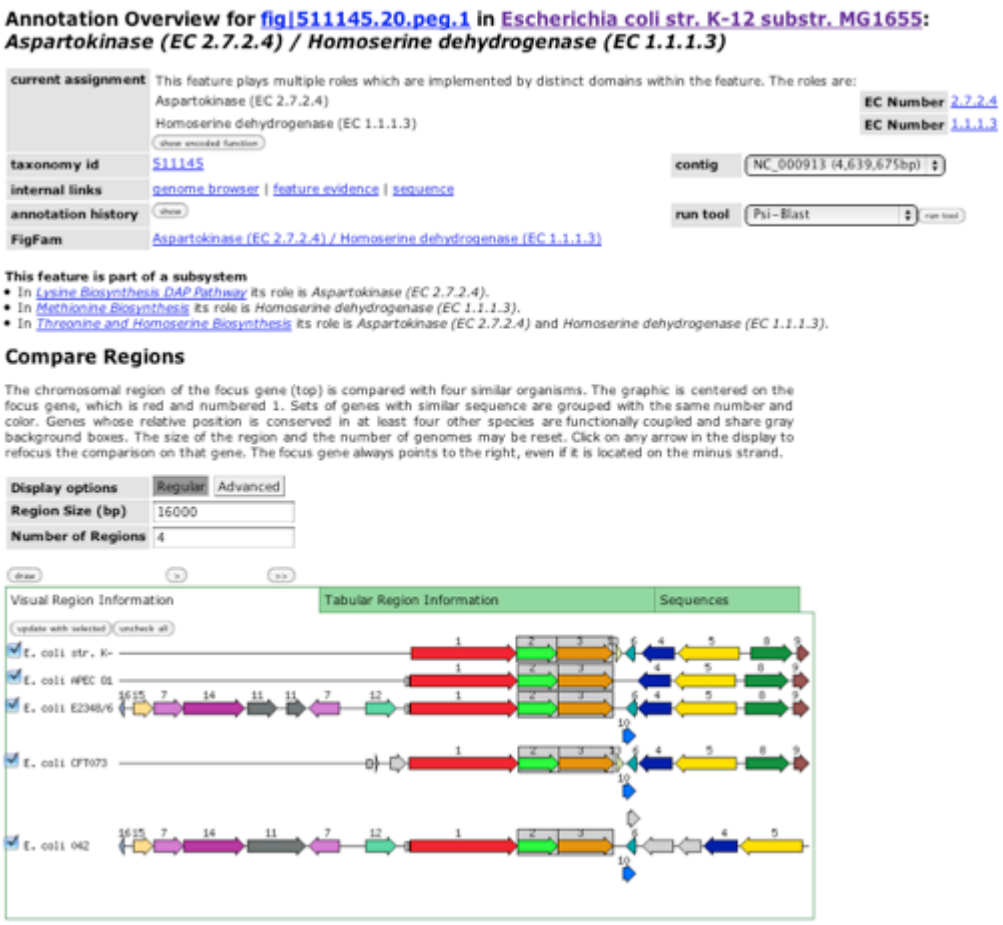
You click on "Browse annotated genome in SEED Viewer" to get started. This brings you to the "Organism Overview Page".



You need to find "Click here to get to the Genome Browser" in the upper right hand box to start the process of looking at your genome.



Go to the first row in the table (the one for peg.1 -- that is, protein-encoding gene 1), and click on the feature ID. This brings you to the "Annotation Overview" page, which is where we will be spending a lot of our efforts.



You should take a little time and study this page. It displays

1. the feature ID,
2. the genome name,
3. the function assigned to the gene product,
4. a history of how the annotation was derived,
5. an EC number (if one is part of the assigned function, the link based on the EC number will be to the KEGG description of the EC),
6. the ability to link to NCBI's Psi-Blast (to get both similarities to known genes and a summary of the recognized domains in the gene product), and (most importantly, we feel)
7. a "compare regions" display that allows you to compare the genes in regions around similar genes in different genomes.

You should explore the links and gain some feel for how to get at the capabilities represented by this page (although there will be many that are beyond the scope of this tutorial).

Now, let us look at the compare regions display in a bit more detail. Note the data that appears for each gene if you hover over it. You should realize that the red gene in the first row of the display is the gene you are "focused on". The other red genes are similar to it and we have attempted to line up corresponding regions from several genomes. You can adjust the size of the regions, or the number of genomes that you wish compared. If you click on the "Advanced" options, you can adjust the threshold used to cluster genes into colors or to find corresponding genes in other genomes (as well as a few other options).

What we are proposing you do now, is move one screen full of compare regions after another, "walking through the genome" to see your annotations and possible errors. This may seem tedious, but for about a day's worth of clicking, you can gain a good sense of the quality and contents of your genome.

To navigate "up" by a full screen, click on ">>" (you can also go left or up by half-screens, but for our purposes, let's go a full screen each time).

Now that you can navigate, let us focus on three important things you can do to change your annotations:

1. If you simply wish to change the annotation of a gene, you can "focus" on that gene, type in the corrected function in "new assignment", and click on change. We suggest that you then click on the "show" button associated with "annotation history" to verify that the change was recorded.
2. If you wish to delete the gene you are focused on, just click on "delete feature", this will delete the gene and refocus you on one of the adjacent genes.
3. Finally, you can insert genes that should have been called, but were not. This operation is nontrivial in this version of RAST. You can do it, and one of our team can walk you through the steps, but for purposes of the tutorial, we will skip that topic (note that it is essentially trivial to add genes in DRAST, and we suggest that technology, once it has matured.

Exporting Your Genome from RAST

Exporting your annotations from RAST is fairly straightforward. You go back to the Organism Overview page, click on "Download" and follow the instructions (see this post showing how)

Good luck, we hope that you do take the time to try our recommended approach, and we hope that it works as well for you as it does for us.

Annotating a Genome with myRAST

It is now possible to get a fairly accurate annotation of a prokaryotic genome in about a day. We honestly believe that the result is often very, very close to what most annotation groups can produce spending months or even man-years. This short tutorial describes our recommended approach to producing a rapid, quite-accurate annotation within about a day (sometimes less for short genomes, and often more for large or diverged genomes).

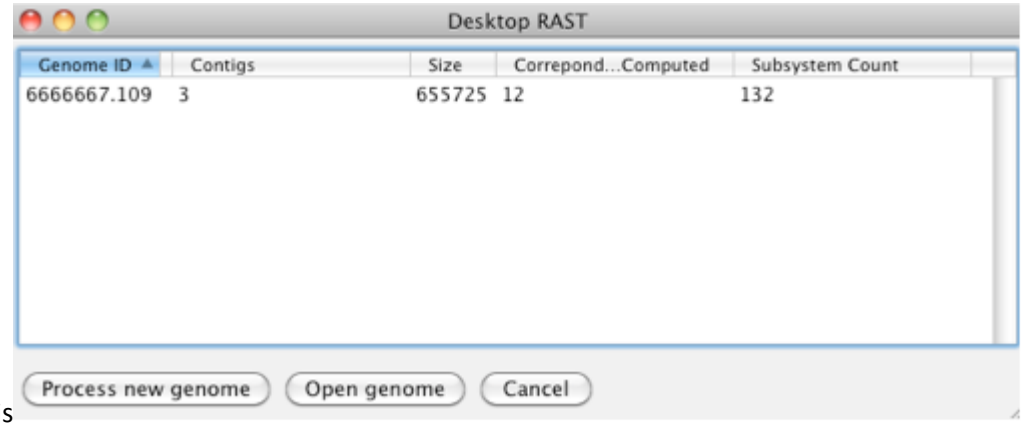
The approach that we advocate is especially suited to annotating a genome that is quite phylogenetically close to an existing (presumably, well-annotated) genome or set of genomes. In particular, it works well for newly sequenced pathogen genomes that are close to large groups of already sequenced genomes.

The proposed approach is as follows:

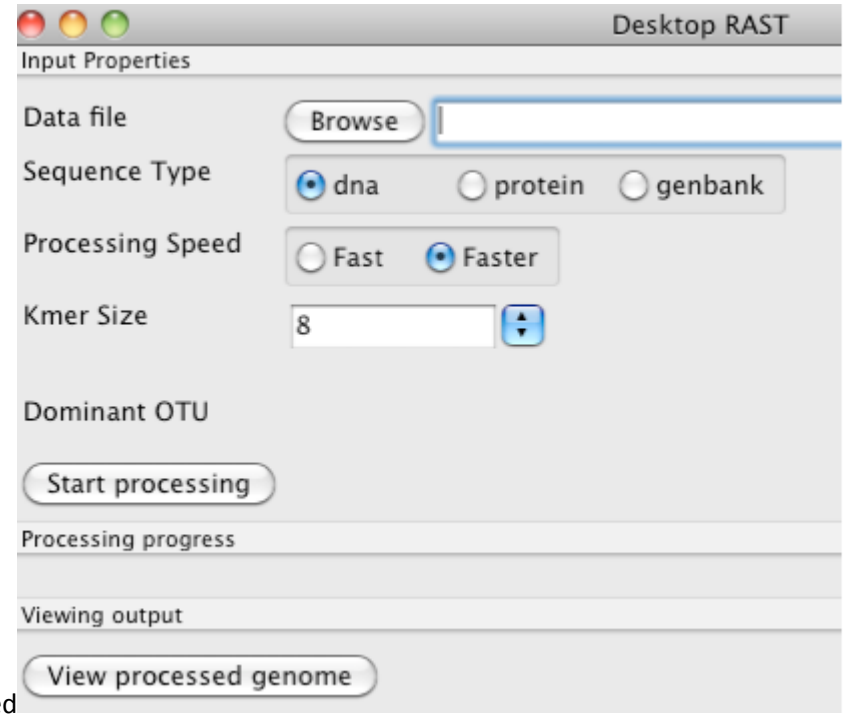
1. Run your genome through myRAST (see ***URL***). This produces an initial annotation. There will probably be errors in gene calls, as well as errors in the assigned functions. Those get cleaned up in the next step.
2. Once you have produced an initial annotation, you can "walk the genome" looking for genes that need to be deleted, inserted, or just re-annotated.
3. Once you have made a quick pass through the genome, we suggest that you export the genome. You will probably wish to do this twice -- once to produce a Genbank formatted version (which can be used by many tools) and a second as a set of tab-separated files suitable for perusing in a tool like Excel.

Running a Genome Through myRAST

Once you start myRAST you should see a screen similar to this



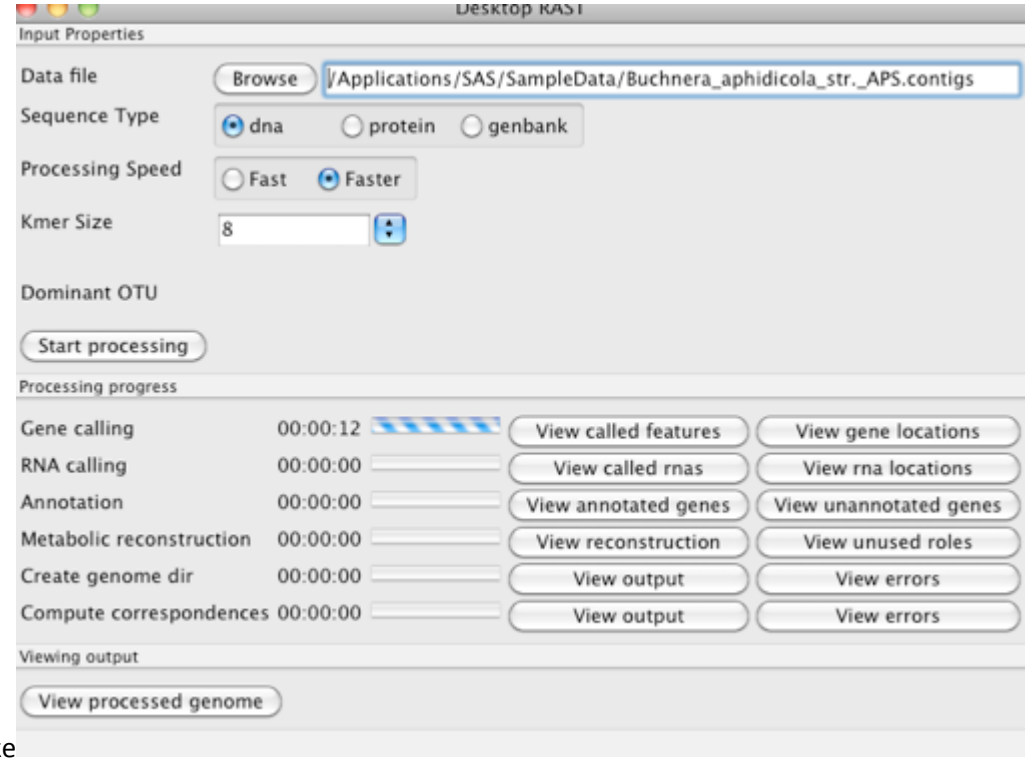
If you click on **Process new genome**, you will be prompted to pick a file containing the genome



to be annotated

You can take as input a file in Genbank format, a file of contigs in FASTA format, or a file of protein sequences in FASTA format. Normally, you would just specify **DNA** meaning that you want to annotate some contigs. You need to **Browse** to get the actual file, and then you click on **Start processing** to begin building the annotations.

Once you start processing, you will see a "control panel" that looks like

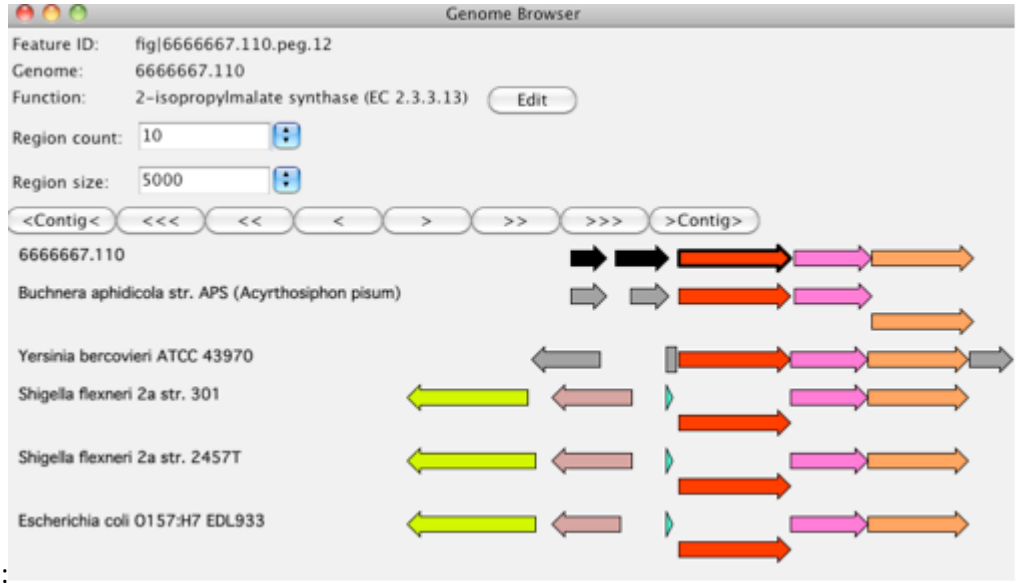


myRAST will go through the annotation steps, you can watch the time it takes, and when it completes you can start perusing the annotations.

Walking your Genome Using myRAST

To begin looking at your annotated genome, you click on **View processed genome.**

The display shows you what we call a "compare regions display":



This display shows a region in your newly-sequenced genome (the first line - in this case Buchnera) along with regions from our collection of annotated genomes. The genes (which we call PEGs for "protein-encoding genes") are colored to make it clear which have the same function. All PEGs with the same color have been annotated with the same function. You should think of yourself as focused on one PEG - the one with the bold outline. Hovering over a PEG will give you at least its ID, the contig containing it, begin and end coordinates, and the function assigned to it. PEGs are depicted as arrows. Other features are depicted as rectangles (e.g., in the Yersinia genome, the leucine operon leader is specified as a 133bp "rna" feature). Quite inconsistently, in the Shigella and E.coli genomes, it was annotated as a 28 aa PEG.

You need to spend a little while just figuring out how to interpret a compare regions display, and then try using the navigate buttons:

- ">" and "<" move you 1 gene to the right or left,
- ">>" and "<<" move you a half screen right or left,
- ">>>" and "<<<" move you a full screen right or left, and
- ">Contig>" and "<Contig<" move you to the beginning of the next or previous contig.

What we are proposing you do now, is move through one screen full of compare regions after another, "walking through the genome" to see your annotations and possible errors. This may seem tedious, but for about a day's worth of clicking, you can gain a good sense of the quality and contents of your genome.

Now that you can navigate, let us focus on three important things you can do to change your annotations:

1. If you simply wish to change the annotation of a gene, you can "focus" on that gene, click the Edit button, and type in the preferred annotation.
2. If you wish to delete the gene, just right-click on the gene and select "Delete feature".
3. Finally, you can insert a gene that should have been called, but was not. To do this, position the cursor on the gap where you think the gene belongs, right-click, and select the intergenic region, and then click on a gene from one of the other genomes that you think corresponds to a missing gene that was not called in the intergenic region. This will cause myRAST to try to find an instance of the annotated PEG in the intergenic region. If it finds it, it will mark it in your newly-sequenced genome.

Now we urge you to spend a while moving through your genome to get a feel for what is there and corrections that you can easily make.

Exporting Your Genome from myRAST

Exporting your annotations from myRAST is fairly straightforward.

There will soon be screenshots here of the procedure.

Good luck, we hope that you do take the time to try our recommended approach, and we hope that it works as well for you as it does for us.

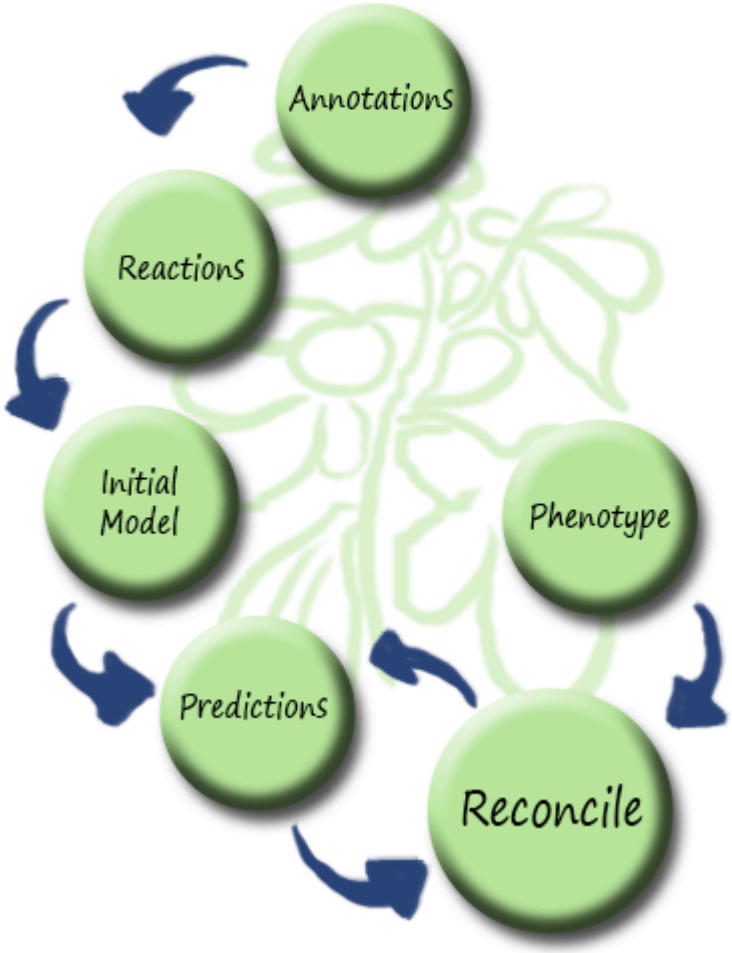
Suggestions on how we could improve the simple set of tools we provide are welcome.

Metabolic Modeling

Modeling Overview

There is an active effort within the SEED Project to make available public metabolic models for many (eventually, thousands) of organisms. Currently, we have a large and growing collection that have already been constructed. You can explore these models and run flux-balance analysis, using a variety of media conditions in the public model-viewer.
You can also build an initial model for any genome that you submit to RAST. Soon, we will support the abilities described in the diagram below, which depicts our suggested steps for using our tools on your newly-sequenced genome.

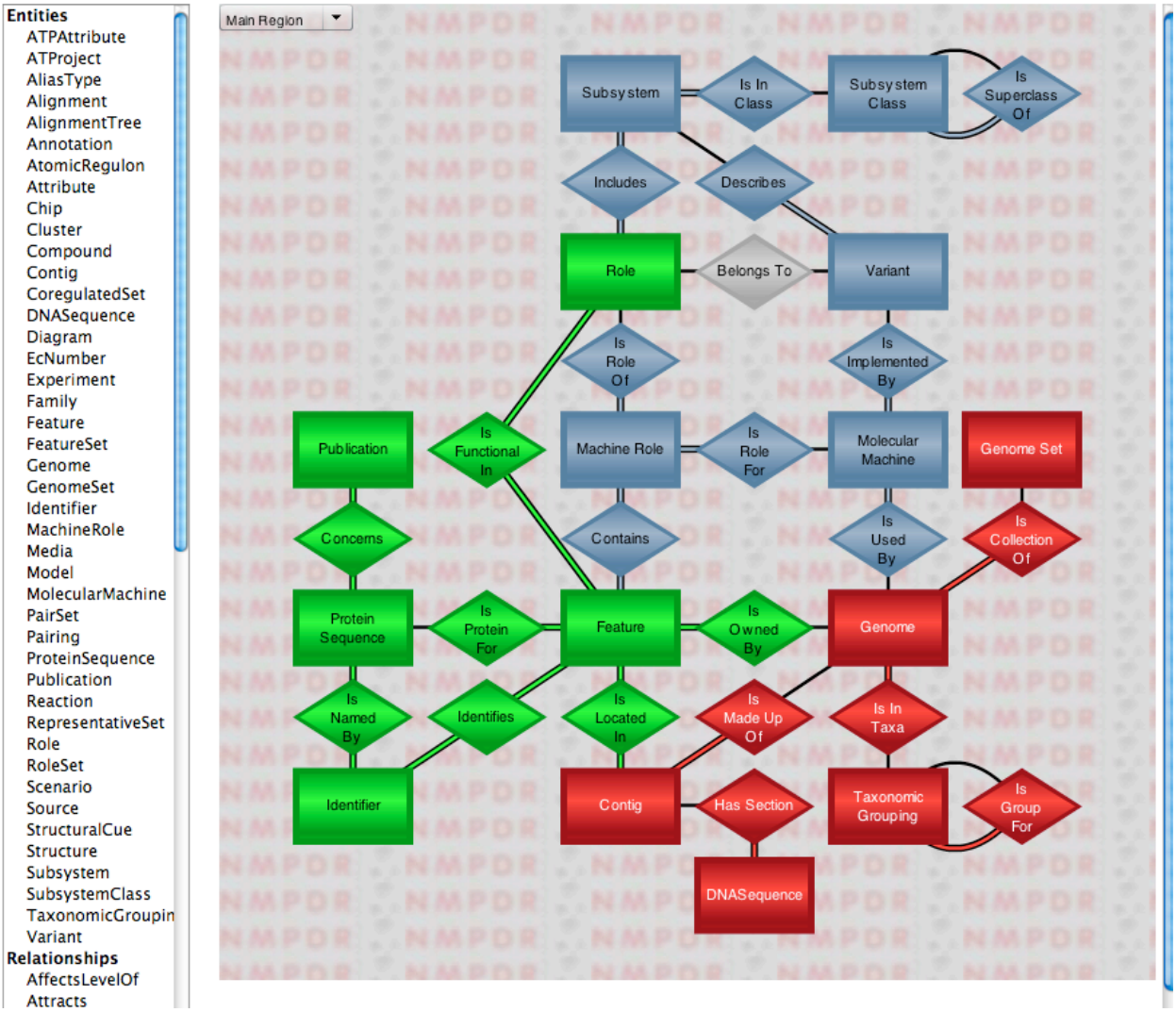
Modeling - The Basic Steps



Accessing The SEED Database

The Entity-Relationship Model

The SEED database is called the Sapling DB. The Sapling DB is described by an entity-relationship model that depicts the basic entities maintained within the database and the relationships that we have encoded between them. This offers the basic foundation upon which most of the SEED toolkit resides. Here is a snapshot of the main RDB page (there are four other pages: Chemistry, Annotations, Models and Expression):



There are programming API's that allow various levels of access to the DB, from writing your own individual data queries to high-level composite operations available via the Perl or

Command Line services, described later. The API's are organized into four "servers", so-called because the DB is accessed through remote server calls.

Summary of Individual Servers

The Sapling Servers API's are released as a set of client-side Perl packages that users can incorporate into their Perl programs to write code to remotely access the Sapling database. They are released as 4 separate packages as described below.

The Sapling Server

The SAPserver.pm (API) package offers programmatic access to the data maintained in the Sapling DB within the SEED. The Sapling DB is described by an entity-relationship model that depicts the basic entities maintained within the database and the relationships that we have encoded between them. This offers the basic foundation upon which most of the SEED toolkit resides. The methods offered by Sapling Objects support a rich set of operations against genomic data. Using the methods described in the API, the user has access to genomes, annotations, functional coupling data, protein families, subsystems, and a rapidly growing number of more specialized forms of data.

To see the overall ER diagram and the relations that implement it see the Sapling webpage.

A complete tutorial is offered in SAP tutorial.

The Annotation Support Server

The ANNOserver.pm (API) package supports capabilities relating to annotation of genomes. It supports invocation of standard gene callers (Glimmer3 for protein-encoding genes), and newly-developed high-performance methods to assign function to protein sequences or regions of DNA fragments (based on FIGfams and a unique use of K-mers that act as signatures of FIGfams). We include an example application based on these methods that can be used to produce relatively accurate annotation of most microbial genomes within a few minutes.

The RAST server

RAST is a publicly-available server for the annotation of microbial genomes. It is maintained by a team at Argonne National Lab and FIG. Currently, it has over 2600 registered users, and several thousand genomes have been run through the service in the last couple of years (often several times!). The RASTserver.pm (API) package was created to support programmatic submission of genomes to RAST, the retrieval of status, and the retrieval of the final set of annotations.

The Model Server

This server (API) provides access to all data associated with the biochemistry database and the genome-scale metabolic models stored within the SEED. This server also provides the user with the ability to run a set of simple flux balance analysis studies with the SEED models. A detailed description of the interface is here.

Getting Started

The SEED servers system is distributed as a small set of Perl packages that the user downloads and installs locally. The distribution can be easily installed on a Mac or Unix-based system, and, soon to be released, on a Windows machine. In addition to the packages that are used in constructing Perl programs, we offer a library of utility programs that offer predefined commands that can be used to extract data from the SEED.

Here are some instructions on how to get started using the servers:

1. Install the distribution: Follow this link for instructions.
2. Try some samples.

Getting started with Command Line "svr" scripts

If you followed the directions carefully, the location of the svr scripts will now be in your path, allowing you to run these from anywhere on your computer.

Reminder: If you are using a Windows machine, use the myRAST Shell, if you are using a mac or linux machine, set this variable in your bash shell

```
export PATH=$PATH://Applications/myRAST.app/bin
```

A list of all available svr scripts and documentation is available here.

List all Genomes

A simple request would be to ask for a list of all genomes in the system. The command *svr_all_genomes* will produce a 2-column table. The first column would contain the names of all genomes, and the second the IDs of those genomes.

Here is a sample of the output from this command:

```
> svr_all_genomes
Berardius bairdii     48742.1
Simian immunodeficiency virus    11723.1
Erythrobacter litoralis HTCC2594 314225.3
Bacteriophage N15     40631.1
Bacillus cereus plasmid pPER272  1396.18
Cyanophage P-SSP7     268748.3
Enterococcus faecium plasmid pEF1     1352.12
Lactococcus lactis subsp. lactis Il1403     272623.1
.
.
.
```

List the features for a genome

Use the command *svr_all_features* to list the features for a given genome. The genome id is a command line argument. Here is an example of using this command:

```
> svr_all_features 3702.1 peg
fig|3702.1.peg.1
fig|3702.1.peg.2
fig|3702.1.peg.3
fig|3702.1.peg.4
fig|3702.1.peg.5
fig|3702.1.peg.6
fig|3702.1.peg.7
fig|3702.1.peg.8
fig|3702.1.peg.9
fig|3702.1.peg.10
fig|3702.1.peg.11
fig|3702.1.peg.12
fig|3702.1.peg.13
fig|3702.1.peg.14
fig|3702.1.peg.15
.
.
.
```

Pipe commands together

The command line svr scripts use stdin and stdout to process data and are designed to be piped together when appropriate. For instance, the function *svr_function_of* takes as input a list of feature ids and produces a 2-column table of feature id and function. Here's an example.

```
> svr_all_features 3702.1 peg | svr_function_of
fig|3702.1.peg.1 photosystem II protein D1 (PsbA)
```

```
fig|3702.1.peg.2 maturase
fig|3702.1.peg.3 SSU ribosomal protein S16p, chloroplast
fig|3702.1.peg.4 Photosystem II protein PsbK
fig|3702.1.peg.5 Photosystem II protein PsbI
fig|3702.1.peg.6 ATP synthase alpha chain (EC 3.6.3.14)
fig|3702.1.peg.7 ATP synthase CF0 B chain
fig|3702.1.peg.8 ATP synthase C chain (EC 3.6.3.14)
fig|3702.1.peg.9 ATP synthase CF0 A chain
fig|3702.1.peg.10     SSU ribosomal protein S2p (SAe),
chloroplast
fig|3702.1.peg.11     DNA-directed RNA polymerase delta (=
beta'') subunit (EC 2.7.7.6), chloroplast
fig|3702.1.peg.12     DNA-directed RNA polymerase gamma
subunit (EC 2.7.7.6), chloroplast
fig|3702.1.peg.13     DNA-directed RNA polymerase beta
subunit (EC 2.7.7.6), chloroplast
fig|3702.1.peg.14     Cytochrome b6-f complex subunit VIII
(PetN)
fig|3702.1.peg.15     Photosystem II protein PsbM
.
.
.
```

Getting started writing Perl scripts to access the servers

To make getting started writing Perl scripts as easy as possible, we have supplied a command called svr that you should use in place of perl when executing perl scripts. It knows about the location of the svr libraries and allows you to start writing code immediately without regard to setting up your environment. On the other hand, if you have set up your environment as described in the Linux installation instructions, you can use the perl command as usual to run your scripts.

List all Genomes

```
Here is a program to list all genomes.
#!/usr/bin/perl -w
use strict;
use SeedEnv;

my $sapObject = SAPserver->new();
my $genomes = $sapObject->all_genomes();

foreach my $g (sort { $genomes->{$a} cmp $genomes->{$b} }
keys(%$genomes)) {
    print "$g\t$genomes->{$g}\n";
}
```

Notice the line `use SeedEnv;`. This brings in the Seed server environment that includes the packages for all the servers. This is evident in the line

```
my $sapObject = SAPserver->new();
```

which references the package SAPserver even though SAPserver is not explicitly included.

And here is a sample of the output:

```
470865.2        44AHJD-like phages Staphylococcus phage
SAP-2
11788.1 Abelson murine leukemia virus
10815.1 Abutilon mosaic virus
5755.1  Acanthamoeba castellanii
212035.3        Acanthamoeba polyphaga mimivirus
329726.3        Acaryochloris marina MBIC11017 plasmid
pREB1
329726.4        Acaryochloris marina MBIC11017 plasmid
pREB2
329726.6        Acaryochloris marina MBIC11017 plasmid
pREB4
329726.7        Acaryochloris marina MBIC11017 plasmid
pREB5
329726.8        Acaryochloris marina MBIC11017 plasmid
pREB6
329726.9        Acaryochloris marina MBIC11017 plasmid
pREB7
329726.10       Acaryochloris marina MBIC11017 plasmid
pREB8
435.1   Acetobacter aceti
```

Documentation on the calls available for each server is available at SAPserver, Annotation Support Server, RAST Submission Server and Model Server.

More information is available at the Servers Blog.

Using the Command Line Scripts

A (Very) Minimal Introduction to Some Basic Command-Line Tools

Our growing body of examples presumes a basic understanding of how to use the command line from a terminal window. However, many biologists do not have this background. So, let me try to give a very short tutorial on how a biologist might get started at working from a terminal window.

If you are going to work with our tools, you will need to have **myRAST** installed, and you will need to follow the instructions on how to gain access to the **SVR tools** that come with it (see the installation guide on how to get started). The tools we discuss are basic "Unix Tools" in the sense that they were first implemented and distributed in the Unix environment. The myRAST distribution includes open source versions that run in the Windows, Mac and Linux environments. For a really excellent book describing the Unix utilities we discuss, I recommend **Unix Utilities by R. Tare** (the last time I checked you could get used copies for about $1 through Amazon – it is an old book, but very well written).

In my minimal comments here, I assume that you

- know how to bring up a terminal window,
- how to invoke the SVR tools, and
- how to build a file using a text editor.

I realize that these are nontrivial requirements, but they are best handled separately by talking with a friend that can help you get started. There are too many differences between the Windows, Macintosh, and Linux environments for me to cover all of those details here.

When you open a terminal window, you should think of yourself as positioned within your **home directory** (or home folder). You would normally see a prompt, which means that your machine is waiting for you to type in a command. If you use our tools to explore different problems, we suggest that you construct a subdirectory of projects; you could call it **Projects.** To make such a directory (or "folder") you can simply type

```
mkdir Projects
```

mkdir is the Unix command that makes a directory. To move from your current position (in the home directory) to the new directory, you would just type

```
cd Projects
```

The **cd** command changes the directory you are in. There are two arbitrary names that you need to remember:

1. the character ~ means *your home directory,* and
2. the string *".."* means your parent directory

Thus,

`cd ~` moves your position to your home directory

`cd Projects` moves you to the Project subdirectory

`cd ..` moves you back to your home directory

Now, try

```
cd ~
cd Projects
mkdir FunctionalCoupling
cd FunctionalCoupling
```

This should create a new directory *(~/Projects/FunctionalCoupling*) and position you in the new directory. Now create a file

```
~/Projects/FunctionalCoupling/a.peg
```

containing just the line

```
fig|100226.1.peg.5307
```

Now, if you run

```
svr_functionally_coupled < a.peg > fc.pegs
```

you should get the following lines in fc.pegs:

```
fig|100226.1.peg.5307   8         fig|100226.1.peg.5303
fig|100226.1.peg.5307   23       fig|100226.1.peg.5304
fig|100226.1.peg.5307   28       fig|100226.1.peg.5305
fig|100226.1.peg.5307   50       fig|100226.1.peg.5306
fig|100226.1.peg.5307   127     fig|100226.1.peg.5308
```

```
fig|100226.1.peg.5307   28        fig|100226.1.peg.5309
```

This gives you the functionally-coupled PEGs along with a core (the higher the score, the more likely that the PEGs are functionally related). If you were now to run

```
svr_functionally_coupled < a.peg | sort –k 2 –n –r > fc2.sorted
```

it would produce

```
fig|100226.1.peg.5307   127     fig|100226.1.peg.5308
fig|100226.1.peg.5307   50        fig|100226.1.peg.5306
fig|100226.1.peg.5307   28        fig|100226.1.peg.5309
fig|100226.1.peg.5307   28        fig|100226.1.peg.5305
fig|100226.1.peg.5307   23        fig|100226.1.peg.5304
fig|100226.1.peg.5307   8          fig|100226.1.peg.5303
```

in *fc2.sorted.* This is not particularly important when there are just a few lines, but for commands that produce big tables, the ability to easily sort them is critical. The parameters for the sort (as shown) were

- -k 2 [means "sort on the second field in the table"]
- -n [means that the second field is numeric; the default is *not sumeric*]
- -r [means "give the output in *reversed order* – that is highest to lowest]

The **sort** command is used to order the output. If you just want to see lines containing some specific value, use **fgrep**. For example,

```
fgrep 'fig|100226.1.peg.5303' fc2.sorted
```

would produce the single line

```
     fig|100226.1.peg.5307   8       fig|100226.1.peg.5303
```

Using **sort** and **fgrep**, you can reorder and take subsets. Using **head,** you can look at just the initial lines in a file. Thus,

```
svr_functionally_coupled < a.peg | sort –k 2 –n –r | head –n 2
```

would produce

```
fig|100226.1.peg.5307   127     fig|100226.1.peg.5308
fig|100226.1.peg.5307   50      fig|100226.1.peg.5306
```

That is, the output is sorted, and then the first 2 lines are taken to get the best two values. With such a small file, this may seem pointless. However, if you had 100,000 lines of output, the ability to sort it (placing the best values at the front), and then grabbing the best values off the front, becomes extremely useful.

Finally, if you wanted to just see the second and third fields in the table, you would use

```
svr_functionally_coupled < a.peg | sort –k 2 –n –r | cut –f 2,3
```

which would produce

```
127     fig|100226.1.peg.5308
50      fig|100226.1.peg.5306
28      fig|100226.1.peg.5309
28      fig|100226.1.peg.5305
23      fig|100226.1.peg.5304
8       fig|100226.1.peg.5303
```

That is, **cut** is used to extract specific columns from the output table.

Let me summarize:

The Unix commands can be used, along with the SVR scripts to compose sequences of commands to give you precisely what you need.

Use **fgrep** to extract lines from an input file.

Use **sort** to order the lines in the output file.

Use **head** to keep just the start of the file.

Use **cut** to extract columns.

Command Line Services

We supply a number of predefined shell scripts that provide basic bioinformatics functions using the servers. These scripts are all prefaced with "svr_" and are found in the bin directory of the distribution. These are designed to use stdin and stdout and to be piped together to form more complex processing.

If you are a MAC or Linux user, these scripts are accessed from the command line in your terminal shell where you must put myRAST in your path, like this:

```
export PATH=$PATH:/Applications/myRAST.app/bin
```

If you are a windows user, you must use the myRAST shell, which is installed with myRAST.

The svr scripts can be directed to use the SEED or the PSEED by the use of the environmental variable SAS_SERVER. It defaults to the SEED server, but if you want your scripts to access the PSEED, you would set the shell variable SAS_SERVER to PSEED, like this, using bash shell

```
export SAS_SERVER=PSEED
```

or like this if you are a windows user

```
set SAS_SERVER=PSEED
```

As a short example of using these scripts, to get a list of all genomes, you could do this at the command line:

```
svr_all_genomes complete
```

This would produce a two column table of all genomes (all complete genomes if you use the "complete" argument) in the SEED or PSEED. The first column is the genome name, and the second is the id, like this:

```
Berardius bairdii 48742.1
Simian immunodeficiency virus 11723.1
Erythrobacter litoralis HTCC2594 314225.3
Bacteriophage N15 40631.1
Bacillus cereus plasmid pPER272 1396.18
Cyanophage P-SSP7 268748.3
Enterococcus faecium plasmid pEF1 1352.12
Lactococcus lactis subsp. lactis Il1403 272623.1
```

```
Salmonella enterica subsp. enterica serovar Newport str.
SL254 423368.6
Cotton leaf curl Rajasthan virus 223259.1
```

Here are examples of a number of basic functions using the servers that can be run from the command line and piped together to create small systems. These should serve as models for others who wish to create their own custom bioinformatics systems using the servers.

Find all features for a genome.

Simply retrieving all the features for a given genome is often the first step in an analysis sequence. This command is designed to be issued at the command line and takes as arguments a genome id and a feature type. The output is a single column table containing feature ids, suitable for piping into subsequent commands.

```
svr_all_features genome_id feature_type
```

This script returns all features of the specified type for the specified genome.

The code for this server is here. The man page is here.

Here is an example of running this command:

```
> svr_all_features 3702.1 peg
fig|3702.1.peg.1
fig|3702.1.peg.2
fig|3702.1.peg.3
fig|3702.1.peg.4
fig|3702.1.peg.5
fig|3702.1.peg.6
fig|3702.1.peg.7
fig|3702.1.peg.8
fig|3702.1.peg.9
fig|3702.1.peg.10
fig|3702.1.peg.11
fig|3702.1.peg.12
fig|3702.1.peg.13
fig|3702.1.peg.14
fig|3702.1.peg.15
.
.
.
```

Find Gene Function

Given a set of protein-encoding genes, a next step might be to retrieve the assigned function for each gene. This command takes as input a single column table of gene ids and returns a tab-separated two column table of gene id and function.

```
svr_function_of < table_of_gene_ids
```

Note that this script uses stdin and stdout and is designed to be part of a processing pipeline.

The code for this server is here. The man page is here.

Here is an example of running this command:

```
> svr_all_features 3702.1 peg | svr_function_of
fig|3702.1.peg.1 photosystem II protein D1 (PsbA)
fig|3702.1.peg.2 maturase
fig|3702.1.peg.3 SSU ribosomal protein S16p, chloroplast
fig|3702.1.peg.4 Photosystem II protein PsbK
fig|3702.1.peg.5 Photosystem II protein PsbI
fig|3702.1.peg.6 ATP synthase alpha chain (EC 3.6.3.14)
fig|3702.1.peg.7 ATP synthase CF0 B chain
fig|3702.1.peg.8 ATP synthase C chain (EC 3.6.3.14)
fig|3702.1.peg.9 ATP synthase CF0 A chain
fig|3702.1.peg.10     SSU ribosomal protein S2p (SAe),
chloroplast
fig|3702.1.peg.11     DNA-directed RNA polymerase delta (=
beta'') subunit (EC 2.7.7.6), chloroplast
fig|3702.1.peg.12     DNA-directed RNA polymerase gamma
subunit (EC 2.7.7.6), chloroplast
fig|3702.1.peg.13     DNA-directed RNA polymerase beta
subunit (EC 2.7.7.6), chloroplast
fig|3702.1.peg.14     Cytochrome b6-f complex subunit VIII
(PetN)
fig|3702.1.peg.15     Photosystem II protein PsbM
.
.
.
```

Find Gene Aliases

Instead of function, perhaps you wish to see all the aliases by which a given feature or set of features is known in the SEED. You would use this command that behaves just like svr_function_of except it returns aliases:

```
svr_aliases_of < table_of_gene_ids
```

Note that this script uses stdin and stdout and is designed to be part of a processing pipeline.

The code for this server is here. The man page is here.

Here is an example of running this command:

```
> svr_all_features 3702.1 peg | svr_aliases_of
fig|3702.1.peg.1
     gi|112382048,gi|113200888,gi|114054364,gi|114107113,gi
|114329726,gi|115531894,gi|134286292,gi|134286378,
gi|134286553,gi|134286643,gi|134286733,gi|134286999,gi|1393
```

```
87232,gi|139389076,gi|139389398,gi|139389623,gi|139389781,g
i|13938993
1,gi|156597939,gi|156598592,gi|157695865,gi|159792928,gi|15
9793098,gi|159895452,gi|159895537,gi|166344112,gi|167391785
,gi|169142
690,gi|169142840,gi|169142925,gi|169143011,gi|169794053,gi|
6723714,sp|A4QJR4,sp|A4QJZ9,sp|A4QKH2,sp|A4QKR1,sp|A4QL00,s
p|A4QLR3,s
p|B0Z4K6,sp|B0Z4U0,sp|B0Z524,sp|B0Z5A8,sp|B1A915,sp|B1NWD0,
sp|P83755,sp|P83755,sp|P83756,sp|P83756,sp|Q06FY1,sp|Q09G66
,sp|Q0G9Y2
,tr|A4QJZ9,tr|A4QKH2,tr|A4QL00,tr|A4QLR3,tr|A9QAZ4,tr|A9QAZ
4,tr|A9QBW0,tr|A9QBW0,tr|Q06FY1,tr|Q09G66,tr|Q0G9Y2,fig|370
2.1.peg.1,
gi|515374,gi|5881674,gi|7525013,fig|85636.1.peg.1,gi|135182
99
fig|3702.1.peg.2
     gi|12002371,gi|12002415,gi|12002417,gi|12002419,gi|120
02421,gi|12002423,gi|12002425,gi|12002427,gi|12002
429,gi|12002431,gi|126022795,gi|5881675
fig|3702.1.peg.3 sp|P56806,sp|P56806,gi|5881676,gi|7525015
fig|3702.1.peg.4
     sp|P56782,sp|P56782,fig|3702.1.peg.4,gi|5881677,gi|752
5016
.
.
.
```

Find Neighbors

Beyond the basics of finding aliases and function, a more advanced analysis might require finding the PEGs that are in the neighborhood of a given PEG. This command takes as input a tab-separated table where the last field in each line contains the PEG for which a list of neighbors is being requested. It takes an argument telling how many neighbors to find to the left and right. The output file is the input file with an extra column appended at the end (containing a list of neighbors).

```
svr_neighbors_of n < table_of_gene_ids
```

Note that this script uses stdin and stdout and is designed to be part of a processing pipeline.

The code for this server is here. The man page is here.

Here is an example of running this command:

```
> svr_all_features 3702.1 peg | svr_neighbors_of 5
fig|3702.1.peg.1
     fig|3702.1.peg.2,fig|3702.1.peg.3,fig|3702.1.peg.4,fig
|3702.1.peg.5,fig|3702.1.peg.6
fig|3702.1.peg.2
     fig|3702.1.peg.1,fig|3702.1.peg.3,fig|3702.1.peg.4,fig
```

```
|3702.1.peg.5,fig|3702.1.peg.6,fig|3702.1.peg.7
fig|3702.1.peg.3
      fig|3702.1.peg.1,fig|3702.1.peg.2,fig|3702.1.peg.4,fig
|3702.1.peg.5,fig|3702.1.peg.6,fig|3702.1.peg.7,fi
g|3702.1.peg.8
fig|3702.1.peg.4
      fig|3702.1.peg.1,fig|3702.1.peg.2,fig|3702.1.peg.3,fig
|3702.1.peg.5,fig|3702.1.peg.6,fig|3702.1.peg.7,fi
g|3702.1.peg.8,fig|3702.1.peg.9
fig|3702.1.peg.5
      fig|3702.1.peg.1,fig|3702.1.peg.2,fig|3702.1.peg.3,fig
|3702.1.peg.4,fig|3702.1.peg.6,fig|3702.1.peg.7,fi
g|3702.1.peg.8,fig|3702.1.peg.9,fig|3702.1.peg.10
.
.
.
```

More Examples

For more examples on how to use the svr_commands, please see

- Downloading a Subsystem
- Downloading a Genome
- Downloading the FIGfams
- Getting Summaries of Functional Content and OTUs for an Metagenomic Sample
- An Etude Relating to a Metagenomics Sample
- Annotating a genome using the SEED servers
- A Short Note on Mapping PEGs to Subsystems
- A Short Note on Use of Server Scripts to Access Functional Coupling Scores
- Downloading the FIGfams
- Getting all IDs, Aliases and Assertions of Function for One or more Protein sequences

The RAST Batch Interface

With the servers, we have supplied a set of RAST batch scripts. Each of these RAST scripts (except for the submission script, which has much more complex arguments) takes as the first two arguments the username and password of a valid account on the RAST server. A single job may be submitted using svr__submit_RAST_job. This script takes a number of arguments that define the parameters for the submission:

--user username	RAST login for the submitting user
--passwd password	RAST password for the submitting user
--genbank filename	If submitting a genbank file, the file of input data.
--fasta filename	If submitting a FASTA file of contigs, the file of input data.
--domain Bacteria or	
--domain Archaea	Domain of the submitted genome.
--taxon_id taxonomy-id	The NCBI taxonomy id of the submitted genome
--bioname "genus species str."	Biological name of the submitted genome
--genetic_code (11 | 4)	Genetic code for the submitted genome, either 11 or 4.
--gene_caller	Gene caller to use (FigFam-base RAST gene caller or straight Glimmer-3)
--reannotate_only	Preserve the original gene calls and use RAST

A set of jobs may be submitted using svr_run_RAST_jobs. This script takes a file of Genbank contig ids on the standard input, and will determine all Genbank contigs that are part of the same Genbank project, and submit them as a single RAST job.

```
svr_run_RAST_jobs username password < contig-ids
```

The script will return information and statistics about the job submission, as well as a job number.

The status of one or more jobs may be checked using svr_status_of_RAST_job:

```
svr_status_of_RAST_job username password jobnumber jobnumber ...
```

When it returns a status of "completed", the job is ready for retrieval.

The results of a job may be retrieved using svr_retrieve_RAST_job. The supported formats are as follows.

genbank	(Genbank format)
genbank_stripped	(Genbank with EC numbers removed)
embl	(EMBL format)
embl_stripped	(EMBL with EC numbers stripped)
gff3	(GFF3 format)
gff3_stripped	(GFF3 with EC numbers stripped)
rast_tarball	(gzipped tar file of the entire job)

```
svr_retrieve_RAST_job username password jobnumber format > output-file
```

If necessary, an executing job may be killed:

```
svr_kill_RAST_job username password jobnumber
```

And deleted from the system entirely (careful, this may not be undone):

```
svr_delete_RAST_job username password jobnumber
```

Advanced Programming with the Servers

Getting a list of Genomes and Their Taxonomies

The Sapling Server has several methods for retrieving information about genomes. In this tutorial, we'll discuss how to get a list of all the genomes and pull out basic data and metrics.

Listing All Genomes

The all_genomes method returns a complete list of the genomes in the database, returning a reference to a hash that maps each genome ID to its scientific name. The following code gets the hash and prints it out.

```
   use SeedEnv;

   my $sapObject = SAPserver->new();
   my $genomeHash = $sapObject->all_genomes();
   for my $genomeID (sort keys %$genomeHash) {
      print "$genomeID: $genomeHash->{$genomeID}\n";
   }
```

The initial output from the above program looks like this:

```
100226.1: Streptomyces coelicolor A3(2)
100226.8: Streptomyces coelicolor A3(2) plasmid SCP1
100226.9: Streptomyces coelicolor A3(2) plasmid SCP2
100379.3: Onion yellows phytoplasma NIM plasmid
extrachromosomal DNA
100379.4: Onion yellows phytoplasma plasmid EcOYW1
100379.5: Onion yellows phytoplasma plasmid pOYM
100379.6: Onion yellows phytoplasma plasmid pOYNIM
```

The `-complete` option can be used in the **all_genomes** call to return only complete genomes, as follows.

```
   use SeedEnv;

   my $sapObject = SAPserver->new();
   my $genomeHash = $sapObject->all_genomes(-complete =>
1);
   for my $genomeID (sort keys %$genomeHash) {
      print "$genomeID: $genomeHash->{$genomeID}\n";
   }
```

This also eliminates the plasmids, as you can see from the output fragment below.

```
```

```
100226.1: Streptomyces coelicolor A3(2)
10090.3: Mus musculus (House mouse)
101031.3: Bacillus B-14905
10116.3: Rattus norvegicus (Norway rat)
101510.15: Rhodococcus jostii RHA1
103690.1: Nostoc sp. PCC 7120
106370.11: Frankia sp. Ccl3
```

Taxonomy

Once you have genome IDs, there are numerous things you can do to get more information. The following program prints a full taxonomy for each complete genome in the system.

```
   use SeedEnv;

   my $sapObject = SAPserver->new();
   my $genomeHash = $sapObject->all_genomes(-complete =>
1);
   my $taxHash = $sapObject->taxonomy_of(-ids => [keys
%$genomeHash]);
   for my $genomeID (sort keys %$genomeHash) {
      print "$genomeID: " . join(", ", @{$taxHash-
>{$genomeID}}) . "\n";
   }
```

In the above fragment, the keys of the initial genome hash specify the list of genomes whose taxonomies are desired. The **taxonomy_of** method computes the taxonomies and puts them in `$taxHash` in the form of lists so that they can be printed by the `for` loop. The output looks like this.

```
00226.1: Bacteria, Actinobacteria, Actinobacteria (class),
Actinobacteridae, Actinomycetales, Streptomycineae,
Streptomycetaceae, Streptomyces, Streptomyces coelicolor,
Streptomyces coelicolor A3(2)
10090.3: Eukaryota, Fungi/Metazoa group, Metazoa,
Eumetazoa, Bilateria, Coelomata, Deuterostomia, Chordata,
Craniata, Vertebrata, Gnathostomata, Teleostomi,
Euteleostomi, Sarcopterygii, Tetrapoda, Amniota, Mammalia,
Theria, Eutheria, Euarchontoglires, Glires, Rodentia,
Sciurognathi, Muroidea, Muridae, Murinae, Mus, Mus musculus
101031.3: Bacteria, Firmicutes, Bacilli, Bacillales,
Bacillaceae, Bacillus, Bacillus sp. B-14905
```

If full taxonomy information is excessive, you can ask for just the domain using **genome_domain**.

```
   use SeedEnv;

   my $sapObject = SAPserver->new();
```

```
   my $genomeHash = $sapObject->all_genomes(-complete =>
1);
   my $domHash = $sapObject->genome_domain(-ids => [keys
%$genomeHash]);
   for my $genomeID (sort keys %$genomeHash) {
      print "$genomeID: $domHash->{$genomeID}\n";
   }

100226.1: Bacteria
10090.3: Eukaryota
101031.3: Bacteria
10116.3: Eukaryota
101510.15: Bacteria
103690.1: Bacteria
```

Retrieving Features and Functions for a Genome

Once you have a genome ID, there are numerous Sapling Server methods for processing the genes and features of the genome. In this article, we will show how to get all the protein sequences, features, and annotations of a single genome.

Given a single genome ID, the `all_proteins` method will return a reference to a hash that maps each gene ID from the genome to its protein sequence. The following program lists all the genes and protein sequences for the Campylobacter genome 360108.3.

```
    use SeedEnv;

   my $sapObject = SAPserver->new();
   my $protHash = $sapObject->all_proteins(-id =>
360108.3);
   for my $geneID (sort { by_fig_id($a,$b) } keys
%$protHash) {
      print "$geneID: $protHash->{$geneID}\n";
   }
```

The `by_fig_id` method in the sort clause orders the gene IDs in natural sequence rather than alphabetically. The first few lines of the output look like this.

```
fig|360108.3.peg.1:
MLEFVFIILILGIVFNLGSLYLKKDNLLEGAIQILNDIQYTQSLAMMQEGIRVDELAIA
KREWFKSRWQIYFIKSAATGYDQTYTIFLDKNGDGNANLGKTEINIDREIAVDVINHNK
LMNSGQSGVISKDDEKTTQRFNLTKRFGIEKVEFKGSCSGFTRLVFDEMGRVYSPLKNA
```

```
NYAYEKTLAKNNSDCIIRLLSKKHALCIVIDTLSGYAYIPDFKTLKSQFVNIKNKNYEC
SKI
fig|360108.3.peg.2:
MEKIKNYKLIIILLSLDLLALLYGTSTLSISADEADIYFGEQGKSLIFSYSLLYYISHF
GTFIFGQNDFGLRLPFLFFHFLSCLLLYLLALKYTKTKIDAFFSLLLFVLLPGTVASAL
LINAASLVIFLTLAILCAYEYEKKWLFYILLIMVLFVDKSFNILFLTFFFFGIYKRNAI
LFTLSLVLFGVSISFYGFDTGGRPRGYFLDTLGIFAACFSPLVFVYFFYTIYRLTFQKY
KNLLWFLMSVTFVFCLLLSLRQKLFLDDFLPFCVICTPLLIKTLMQSYRVRLPVFRLRY
KIFIECSIIFLIFCYFLIVANQLLYYFINNPNRHFANNYHFAKELALELKKQDVLELAT
APSLQKRLRFYGIKNSNKFYLKALKQADKYDMDKKIVKVKLGKYEKVYQILNYD
fig|360108.3.peg.3:
MQTIDQIFQTQIDIKKSTFLSFLCPFEDFKFLIETLKKEHPKAVHFVYAYRVLNDFNQI
AEDKSDDGEPKGTSGMPTLNVLRGYDLINAALITVRYFGGIKLGTGGLVRAYSDAANAV
INNSSLLSFELKKNITIAIDLKNLNRFEHFLKTYSFNFTKDFKDCKAILHIKLDEKEEQ
EFEIFCKNFAPFEIEKL
fig|360108.3.peg.4:
MQVNYRTISSYEYDAISGQYKQVDKQIEDYSSSGDSDFMDMLNKADEKSSGDALNSSSS
FQSNAQNSNSNLSNYAQMSNVYAYRFRQNEGELSMRAQSASVHNDLTQQGVNEQSKNNT
LLNDLLNAI
```

If all you want is the list of gene IDs, you can use the `all_features` method. This method allows you to specify multiple genomes (the `-ids` parameter) and what types of gene IDs you want (the `-types` parameter). For example, this program asks for the RNA features in the two Streptococcus pyogenes genomes 160490.1 and 198466.1.

```
    use SeedEnv;

    my $sapObject = SAPserver->new();
    my $rnaHash = $sapObject->all_features(-ids =>
[160490.1, 198466.1],
                                           -type => 'rna');
    for my $genome (keys %$rnaHash) {
        print "RNAs for $genome\n";
        for my $rna (@{$rnaHash->{$genome}}) {
            print "   $rna\n";
        }
    }
```

Given a list of gene IDs, the `ids_to_functions` method can be used to retrieve the functional assignments of the genes. For example, the following program displays the functional assignment for each of the RNAs in the two selected genomes.

```
    use SeedEnv;

    my $sapObject = SAPserver->new();
    my $rnaHash = $sapObject->all_features(-ids =>
[160490.1, 198466.1],
                                           -type => 'rna');
    for my $genome (keys %$rnaHash) {
```

```
        print "RNAs for $genome\n";
        my $rnaData = $sapObject->ids_to_functions(-ids =>
$rnaHash->{$genome});
        for my $rna (sort { by_fig_id($a,$b) } keys
%$rnaData) {
            print "   $rna: $rnaData->{$rna}\n";
        }
    }
```

A fragment of the output for the above program is shown below.

```
RNAs for 198466.1
   fig|198466.1.rna.1: SSU rRNA
   fig|198466.1.rna.2: tRNA-Ala
   fig|198466.1.rna.3: LSU rRNA
   fig|198466.1.rna.4: tRNA-Val
   fig|198466.1.rna.5: tRNA-Asp
   fig|198466.1.rna.6: tRNA-Lys
```

Conversion of Gene and Protein ID's

In example1 we illustrate some basic capabilities that relate to determining the set of IDs attached to specific protein sequences. The program accepts a protein ID as input. The ID may be one of several that are maintained by the SEED, UniProt, RefSeq, KEGG and other groups. The program first accesses all IDs attached to identical protein sequences. This can be a fairly large set in cases in which many very similar genomes have been sequenced.

The program determines the set of IDs associated with proteins that have identical sequence to the input protein ID. We call these sequence-equivalent proteins. The program displays the associated protein sequence, and then it computes a table describing these proteins. There will be at least one row for each of the computed IDs. There will be several rows if multiple groups have used the same ID and attached functional assignments to the ID. The columns in the table are as follows:

1. The first column is the ID of the protein.
2. The second is the genus, species and strain of the associated genome (which we sometimes call "the scientific name of the genome")
3. If we believe that the ID corresponds precisely to the gene associated with the input ID, this field will contain a 1. In this case, we speak of the two IDs as "precisely equivalent". Otherwise, if we only know that this ID is for a protein sequence identical to the sequence designated by the input ID, it will contain 0. In this case, we say "the IDs are sequence-equivalent, but not necessarily precisely equivalent."

4. The fourth column gives the functional assignment associated with the protein ID.
5. The fifth column gives the source of the assignment.
6. If we believe that an expert made this assertion, this field will contain 1. Otherwise, it will contain 0.

The need to map IDs between groups and compare asserted functions of proteins is quite basic. It would be straightforward for any group to write a short CGI script using the capabilities illustrated in example1 that supported connecting protein IDs to literature (via PubMed), to structure data (when present), and so forth. Every annotation team needs this class of functionality.

Example 1 Discussion

With the server packages installed, the code in example1 can be run as follows

```
> perl server_paper_example1.pl "fig|83333.1.peg.145"
```

The work of this routine is in two parts. Here, we use the SAP server to build a hash of identifiers for precisely equivalent genes used in determining the contents of column 3 later on.

```
          my %preciseHash;
          my $precise_assertions_list = $sapObject-
>equiv_precise(-ids => $id);
          $precise_assertions_list =
$precise_assertions_list->{$id};
          if (@$precise_assertions_list > 0) {
            my $inputID = $id;
               for my $precise_assertion
(@$precise_assertions_list) {
                   my ($newID, $function, $source, $expert) =
@$precise_assertion;
                   $preciseHash{$newID} = 1;
               }
```

Here, we again use the SAP server to get all sequence equivalent IDs and produce the output table shown below (truncated for this example).

```
          my $assertions = $sapObject->equiv_sequence(-ids
=> $id);
          my $assertions = $assertions->{$id};
          if (@$assertions < 1) {
              print STDERR "No results found.\n";
          } else {
              for my $assertion (@$assertions) {
                  my ($newID, $function, $source, $expert,
$genomeName) = @$assertion;
                  $genomeName = '' if ! defined $genomeName;
                  my $column3 = ($preciseHash{$newID} ? 1 :
0);
```

```
                print join("\t", $newID, $genomeName,
$column3, $function, $source,
                                    $expert) . "\n";
            }
        }
```

Output Table

```
cmr|NT01SF0150          0       dnaK suppressor protein VC0596
[imported] CMR   0
cmr|NT02EC0154          0       dnaK suppressor protein VC0596
[imported] CMR   0
cmr|NT02SB0136          0       RNA polymerase-binding protein
DksA  CMR   0
cmr|NT02SD0166          0       RNA polymerase-binding protein
DksA  CMR   0
cmr|NT02SF0144          0       dnaK suppressor protein VC0596
[imported] CMR   0
cmr|NT03EC0181          0       DksA homolog    CMR   0
cmr|NT04EC0178          0       dnaK suppressor protein VC0596
[imported] CMR   0
cmr|NT04SF0143          0       RNA polymerase-binding protein
DksA  CMR   0
cmr|NT04SS0162          0       RNA polymerase-binding protein
DksA  CMR   0
cmr|NT10EC0149          0       RNA polymerase-binding protein
DksA  CMR   0
.
```

Metabolic Reconstructions Provided for Complete Prokaryotic Genomes

Given a set of functional roles, one often wishes to understand what subsystems can be inferred from the set. This example reads as input a set of functional roles and constructs a table of subsystems, along with their variation codes, that can be identified. The data displayed in this simple example could form the start of a research project to gather the functional roles not connected to subsystems, to determine whether they were not connected because a small set of functional roles were not present in the input, and to seek candidates for such "missing functional roles". The ability to easily map functional roles into subsystems will improve over time, as the SEED annotation effort improves its collection of encoded subsystems [PMID: 16214803].

Example 2 Discussion

With the server packages installed, the code in example 2 can be run as follows:

```
> perl server_paper_example2.pl < Buchnera_roles.tbl
```

The work of this routine is in two parts. First, we use the Subsystem Server to construct the list of role-id pairs for use later on.

```
my $sapObject = SAPserver->new();
my @pairs;
while () {
    chomp;
    my ($id, $role) = split(/\t/, $_);
    push @pairs, [$role, $id];
}
```

Then, we call the Subsystem Server method metabolic reconstruction and process the results to produce the output table.

```
my $reconstruction =
    $ssObject->metabolic_reconstruction(-roles =>
\@pairs);
for my $record (@$reconstruction) {
    my ($variantID, $role, $id) = @$record;
    my ($subsysID, $code) = $variantID =~
/^(.+):(.+)$/;
    print join("\t", $subsysID, $code, $role, $id)
. "\n";
}
```

Example 2 Input File (Truncated)

```
fig|107806.1.peg.1      Anthranilate synthase, aminase
component (EC 4.1.3.27) # TrpAa          82
fig|107806.1.peg.2      Anthranilate synthase,
amidotransferase component (EC 4.1.3.27) # TrpAb Buchnera
aphidicola str. APS (Acyrthosiphon pisum)      167
fig|107806.1.peg.3      Anthranilate synthase,
amidotransferase component (EC 4.1.3.27) # TrpAb Buchnera
aphidicola str. APS (Acyrthosiphon pisum)      167
fig|107806.1.peg.7      2-isopropylmalate synthase (EC
2.3.3.13)        Buchnera aphidicola str. APS
(Acyrthosiphon pisum)      508
fig|107806.1.peg.8      3-isopropylmalate dehydrogenase (EC
1.1.1.85)   Buchnera aphidicola str. APS (Acyrthosiphon
pisum)      353
fig|107806.1.peg.9      3-isopropylmalate dehydratase large
subunit (EC 4.2.1.33)       Buchnera aphidicola str. APS
(Acyrthosiphon pisum)      462
fig|107806.1.peg.10     3-isopropylmalate dehydratase small
subunit (EC 4.2.1.33)              28
fig|107806.1.peg.11     tRNA uridine 5-
carboxymethylaminomethyl modification enzyme GidA
Buchnera aphidicola str. APS (Acyrthosiphon pisum)      150
fig|107806.1.peg.12     ATP synthase A chain (EC 3.6.3.14)
Buchnera aphidicola str. APS (Acyrthosiphon pisum)      264
.
```

```
.
.
```

Example 2 Output Table (Truncated)

```
tRNA processing 1    Ribonuclease P protein component (EC
3.1.26.5)  fig|107806.1.peg.24
tRNA processing 1    tRNA(Ile)-lysidine synthetase
     fig|107806.1.peg.111
tRNA processing 1    tRNA-specific adenosine-34 deaminase
(EC 3.5.4.-)    fig|107806.1.peg.247
tRNA processing 1    tRNA delta(2)-isopentenylpyrophosphate
transferase (EC 2.5.1.8)    fig|107806.1.peg.541
tRNA processing 1    tRNA-i(6)A37 methylthiotransferase
     fig|107806.1.peg.421
tRNA processing 1    Ribonuclease T (EC 3.1.13.-)
     fig|107806.1.peg.187
tRNA processing 1    tRNA pseudouridine synthase B (EC
4.2.1.70)  fig|107806.1.peg.361
tRNA processing 1    tRNA pseudouridine synthase A (EC
4.2.1.70)  fig|107806.1.peg.198
tRNA aminoacylation, Arg   1.00  Arginyl-tRNA synthetase (EC
6.1.1.19)  fig|107806.1.peg.239
Experimental-EFP 1    Translation elongation factor P
     fig|107806.1.peg.29
mnm5U34 biosynthesis bacteria    1    tRNA uridine 5-
carboxymethylaminomethyl modification enzyme GidA
     fig|107806.1.peg.11
mnm5U34 biosynthesis bacteria    1    tRNA 5-
methylaminomethyl-2-thiouridine synthase TusD
     fig|107806.1.peg.507
mnm5U34 biosynthesis bacteria    1    tRNA 5-
methylaminomethyl-2-thiouridine synthase TusA
     fig|107806.1.peg.427
mnm5U34 biosynthesis bacteria    1    tRNA 5-
methylaminomethyl-2-thiouridine synthase TusC
     fig|107806.1.peg.506
mnm5U34 biosynthesis bacteria    1    GTPase and tRNA-U34
5-formylation enzyme TrmE  fig|107806.1.peg.26
mnm5U34 biosynthesis bacteria    1    tRNA (5-
methylaminomethyl-2-thiouridylate)-methyltransferase (EC
2.1.1.61)  fig|107806.1.peg.253
mnm5U34 biosynthesis bacteria    1    Cysteine desulfurase
(EC 2.8.1.7)    fig|107806.1.peg.568
mnm5U34 biosynthesis bacteria    1    tRNA 5-
methylaminomethyl-2-thiouridine synthase TusB
     fig|107806.1.peg.505
Ribosome LSU bacterial     1    LSU ribosomal protein L18p
(L5e) fig|107806.1.peg.483
Ribosome LSU bacterial     1    LSU ribosomal protein L11p
(L12e)     fig|107806.1.peg.47
Ribosome LSU bacterial     1    LSU ribosomal protein L23p
```

```
(L23Ae)    fig|107806.1.peg.497
.
```

Locating Functionally Coupled PEGs

By Ross Overbeek (March 3, 2011)

Searching for functionally-related genes using clues supplied by comparative analysis has long been one of our key goals. This feature has been in our toolkits as far back as the early WIT systems. Let me briefly summarize some of our current tools that exist within the SEED server scripts to support this type of analysis.

Our tools read input files that we think of as tab-separated tables. Each tool takes a table as input and writes out a table with extra columns.

To get started, create a file called *a.peg* with the ID of a single PEG in it:

```
fig|83333.1.peg.4
```

This is a gene in *E.coli*. To see the function of the PEG, just run

```
svr_function_of < a.peg
```

You should get

```
fig|83333.1.peg.4       Threonine synthase (EC 4.2.3.1)
```

as output. Notice that an extra column has been added containing the function of the PEG. Had the input file contained more PEGs, you would have gotten equally more lines of output.

To see which genes tend to co-occur on the chromosome with the PEG, run

```
svr_functionally_coupled < a.peg
```

This should produce

```
fig|83333.1.peg.4       57       fig|83333.1.peg.2
fig|83333.1.peg.4       127     fig|83333.1.peg.3
fig|83333.1.peg.4       10       fig|83333.1.peg.6
fig|83333.1.peg.4       10       fig|83333.1.peg.7
fig|83333.1.peg.4       11       fig|83333.1.peg.8
fig|83333.1.peg.4       7         fig|83333.1.peg.9
```

The first line may be taken to say “fig|83333.1.peg.4 occurs close to fig|83333.1.peg.2 and corresponding pairs occur in genomes from 57 distinct OTUs” (here, you may think of an OTU as a collection of genomes that are quite similar; normally, they have ribosomal rRNAs that are closer than 97% identical).

This means that the corresponding pairs of genes occur close to one another in at least 57 genomes that are not really close. If you wanted to see the functions, you would use

```
svr_functionally_coupled < a.peg | svr_function_of
```

which produces

```
fig|83333.1.peg.4       57      fig|83333.1.peg.2       Aspartokinase (EC 2.7.2.4) /
Homoserine dehydrogenase (EC 1.1.1.3)
fig|83333.1.peg.4       127     fig|83333.1.peg.3       Homoserine kinase (EC 2.7.1.39)
fig|83333.1.peg.4       10      fig|83333.1.peg.6       UPF0246 protein YaaA
fig|83333.1.peg.4       10      fig|83333.1.peg.7       Putative alanine/glycine
transport protein
fig|83333.1.peg.4       11      fig|83333.1.peg.8       Transaldolase (EC 2.2.1.2)
fig|83333.1.peg.4       7       fig|83333.1.peg.9       Molybdopterin biosynthesis Mog
protein, molybdochelatase
```

This command would lead you to genes that are close to the input gene and tend to co-occur. But, suppose that you wanted to find genes that might not be close in the given genome, but do tend to co-occur in clusters in lots of other genomes. To find those, you might use something like,

```
svr_ids_to_figfams  < a.peg | svr_fc_figfams -MinSc 100 |
svr_figfams_to_ids 83333.1
```

which would produce

```
fig|83333.1.peg.4       Threonine synthase (EC 4.2.3.1) FIG000134       339     FIG000582
fig|83333.1.peg.3

fig|83333.1.peg.4       Threonine synthase (EC 4.2.3.1) FIG000134       179     FIG000885
fig|83333.1.peg.2

fig|83333.1.peg.4       Threonine synthase (EC 4.2.3.1) FIG000134       179     FIG000885
fig|83333.1.peg.3859
```

The first line may be taken to say that FIGfam FIG000134 and FIGfam FIG000582 tend to co-occur, fig|83333.1.peg.4 is in FIG000134, and fig|83333.1.peg.3 is in FIG000582. I have found that using a '–MinSc 100' seems to work well, but I suggest that you experiment. Note that much lower values produce co-occurrences that are not particularly useful.

We have expression data for *E.coli* (the genome 83333.1 is *Escherichia coli K12*). Hence, we could use

```
svr_corr_by_exp < a.peg
```

which should produce

```
fig|83333.1.peg.4       0.923   fig|83333.1.peg.3
fig|83333.1.peg.4       0.950   fig|83333.1.peg.2
```

The first line says "The expression values for fig|83333.1.peg.4 and fig|83333.1.peg.3 have a Pearson Correlation Coefficient of 0.923". In this case, this only supports things we already knew from chromosomal clustering, but this is often not the case.

To explore the use of expression data a bit further we could ask for genes that do not necessarily have high Pearson Coefficients in this genome, but the corresponding genes in other genomes with expression data show clear correlation. To experiment with this, you might try

```
svr_coregulated_by_correspondence < a.peg
```

which produces a number of lines, beginning with

```
fig|83333.1.peg.4       fig|198094.1.peg.1803,0.875,fig|198094.1.peg.2380
fig|83333.1.peg.1425

fig|83333.1.peg.4       fig|100226.1.peg.5307,0.818,fig|100226.1.peg.5305
fig|83333.1.peg.2791

fig|83333.1.peg.4       fig|1140.3.peg.2694,0.951,fig|1140.3.peg.1897
fig|83333.1.peg.3063
```

.

These lines say that fig|83333.1.peg.4 may be functionally related to fig|83333.1.peg.1425, fig|83333.1.peg.2791, and/or fig|83333.1.peg.1897. In each case, the corresponding genes from another genome, along with the Pearson Correlation Coefficient, are shown (and you may wish to look at those genes to see if this makes sense).

In these brief examples we have shown just the basic tools that can be used to look for functionally related genes. We continue to build more and to use them to pursue clues (which inevitably leads to requests for more). As the growth in the number of available genomes accelerates and the available expression data increases, these tools increase in utility.

Using the Servers for Genome Annotation

Annotating a genome using the SEED servers (via the command line or Perl code)

Suppose that you have a newly sequenced bacterial or archaeal genome. You have a number of options if you want to get a rapid, fairly accurate annotation:

1. You can run the genome through RAST, which will give you an attempt at a comprehensive annotation that is browsable.
2. You can use our Mac-App or PC-App to get a very rapid estimate.
3. You can run a few command line scripts that call the RNA-encoding genes and protein-encoding genes, asserts functions for the gene products, and produces an initial metabolic reconstruction.
4. You can programmatically call the RNAs, PEGs, identify the functions, and generate an initial metabolic reconstruction.

This short tutorial will describe the last two options.

Calling RNA-Encoding Genes Using a Command Line Script

Suppose that you have the contigs of a newly-sequence genome in a file called DNA.fasta. You need to know the genus, species, and whether the genome is bacterial or archaeal. To be concrete, suppose that the file contigs contains the genome of a Buchnera aphidicola (which is a bacteria). Then,

```
   svr_find_rnas Buchnera aphidicola bacteria < contigs >
RNA.fasta 2> RNA.tbl
```

will produce a fasta file of the DNA encoding the RNA genes written to RNA.fasta, and a 5-column tab-separated table describing the location and function of each of the identified genes written to RNA.tbl. Niels Larsen originally coded this tool and we thank him for making it available.

To identify the protein-encoding genes (PEGs) and get the translations of the identified genes, one would use

```
   svr_call_genes < contigs > translations.fasta 2> PEG.tbl
```

This produces a 4-column tab-separated table containing

1. an arbitary ID of the form "prot_nnnnn"

2. the contig containing the gene
3. the position of the first character of the gene
4. the position of the last character of the gene

If the beginning position is less than the ending position, the gene is on the '+' strand; otherwise, it is on the '-' strand. The coordinates include both the start and stop codons, unless the gene runs off the end of a contig. The genes are called by Glimmer3 [see http://www.cbcb.umd.edu/software/glimmer/], and again we are thankful for being able to use such a fine tool (yes, eventually there will be better gene caller's, but Glimmer was certainly a major contribution to the community).

Note that you do not have functions for the protein-encoding genes.

To get those, you might use

```
   svr_assign_using_figfams < translations.fasta >
estimates.of.function 2> failed.to assign.function
```

This will write a 3-column tab-separated table. The columns are as follows:

1. A "score" that relates to reliability. The higher, the more reliable. Values usually range from 1 to several hundred. We suggest that you run some benchmark genomes through (genomes for which you have the annotations),and you can acquire some feel for the correspondence of reliability and score.
2. An ID from the input file.
3. The function we would assign to the protein.

This will fairly rapidly assign functions to PEGs that clearly match one of our protein families, and it will tell you which translations could not be accurately assigned a function using FIGfams. If you want the server to try to assign a function to all translations (assigning "hypothetical protein" in cases it could not make any other assignment) add the -all option:

```
   svr_assign_using_figfams -all < translations.fasta >
estimates.of.function
```

Finally, to place the PEGs into subsystems, you can use

```
   svr_metabolic_reconstruction < estimates.of.function >
metabolic.reconstruction 2> pegs.not.used.in.reconstruction
```

This will write two files. the file metabolic reconstruction will contain input lines that can be placed in subsystems. Each line will have two appended columns: the subsystem name and the variant code. The second file, pegs.not.used.in.reconstruction, will contain the lines from the input file corresponding to PEGs that could not be placed into any subsystem.

Accessing Annotation Services from a Perl Program

The actual Perl code to perform the steps illustrated above is fairly straightforward. We will cover each of the steps using short snippets of Perl code.

First, we recommend starting with something like

```
use SeedEnv;
use ANNOserver;
my $annO = ANNOserver->new;
.
.
.
```

Then, the following code can be used to scan the contents of the file $contigsF. It writes out the fasta sequences of the detected RNAs to the file $RNAseqs, and writes a description of where the genes are located in the file $RNAtbl.

```
open(CONTIGS,"<$contigsF") || die "could not open
$contigsF";
my $results          = $annO->find_rnas( -input   =>
\*CONTIGS,
  -genus   => $genus,
  -species => $species,
  -domain  => $domain
                                       );

open(RNASEQS,">$RNAseqs") || die "could not open $RNAseqs";
open(RNATBL,">$RNAtbl")   || die "could not open $RNAtbl";
my($fasta,$genes) = @$results;
print RNASEQS $fasta;
close(RNASEQS);
foreach $_ (sort { $a->[0] cmp $b->[0] } @$genes)
{
    print RNATBL join("\t",@$_),"\n";
}
close(RNATBL);
close(CONTIGS);
```

To locate the protein-encoding genes, write their translations to the file $PEGtranslations, their locations to $PEGtbl, and the functions that could be assigned to $functions.

```
open(TRANS,">$PEGtranslations") || die "could not open
$PEGtranslations";
open(PEGTBL,">$PEGtbl")         || die "could not open
$PEGtbl";
open(FUNCTIONS,">$functions")   || die "could not open
$functions";
open(CONTIGS,"<$contigsF")      || die "could not open
$contigsF";

$results = $annO->call_genes( -input => \*CONTIGS );
```

```
($fasta,$genes) = @$results;
print TRANS $fasta;
close(TRANS);

foreach $_ (sort { $a->[0] cmp $b->[0] } @$genes)
{
    print PEGTBL join("\t",@$_),"\n";
}
close(PEGTBL);

open(TRANS,"<$PEGtranslations") || die "could not open
$PEGtranslations";
my $handle = $annO->assign_function_to_prot( -input =>
\*TRANS,
      -assignToAll => 1,
      -kmer => 8,
      -scoreThreshhold => 3,
      -hitThreshhold => 3,
      -normalizeScores => 0,
      -detailed => 0
                                              );
my @roles = ();
while (my $tuple = $handle->get_next())
{
    my($peg,$function,undef,$score) = @$tuple;
    foreach my $role
(&SeedUtils::roles_of_function($function))
    {
 push(@roles,[$role,$peg]);
    }
    print FUNCTIONS
join("\t",($score,$peg,$function)),"\n";
}
close(TRANS);
close(FUNCTIONS);
```

Finally, the code to produce the metabolic reconstruction estimate from the list of [$role,$peg] pairs would be as follows:

```
open(MR,">$metabolic_recon") || die "could not open
$metabolic_recon";
my $recon = $annO->metabolic_reconstruction( -roles =>
\@roles );
foreach $_ (@$recon)
{
    print MR join("\t",@$_),"\n";
}
close(MR);
```

These three snippets of code are nontrivial, but when you realize that they actually support

1. the calling of genes,

2. assignment of functions to the genes, and
3. the derivation of a metabolic reconstruction,

we hope that you agree that they are actually quite minimal.

Extending the ends of the contigs.

By: Rob Edwards, 2/12/11

Abstract

We are going to be sequencing a lot of draft genomes, so we need to take a look at automating identification of gaps. This short white paper considers the P. salmonis genome sequenced recently, and demonstrates an approach to identifying gaps and suggesting the length of those gaps. This approach should be automatable with little effort.

Sequence Quality

To start with, I wanted to check the sequence assembly quality. N50 is a pretty easy measure of sequence quality, and the others are simple statistics:

(The N50 is the largest entity E such that at least half of the total size of the entities is contained in entities larger than E. i.e. the median length of the contigs).

We have two versions of the sequence:

RAST job number 16097:

Number of contigs:	1331
Total length:	1,947,012
Average length:	1,462.82
Shortest sequence:	contig01327 (100 bp)
Longest sequence:	contig00001 (32,564 bp)
N50:	3,218

RAST job number 18181:

Number of contigs:	819
Total length:	1,813,314
Average length:	2,214.06
Shortest sequence:	contig00627 (502 bp)
Longest sequence:	contig00077 (32,602 bp)
N50:	3,603

The latter was used for all subsequent work.

End locations

I located the ends of the sequences. Using the SEED servers (http://servers.theseed.org/) I ran the following:

```
svr_just_ends < contigs | svr_assign_to_dna_using_figfams |
svr_possible_joins | strengthen_possible_joins  >
possible_joins.txt
```

And then cleaned up the output a bit to remove some duplicate sequences (the svr_possible_joins code has a minor bug…). Basically the way this works is taking the ends of the sequences, looking for ORFs at the ends, and then listing all cases where there are different contigs that have the same protein on the end. These are candidate joins, and are shown in the table below.

Contigs	Start	Stop	Shared Function	Note
contig00973	88	243	4-hydroxyphenylpyruvate dioxygenase (EC 1.13.11.27)	
contig00423	1134	1799	4-hydroxyphenylpyruvate dioxygenase (EC 1.13.11.27)	
contig00102	1355	15	ABC transporter, ATP-binding protein	1
contig00084	804	229	ABC transporter, ATP-binding protein	1
contig00033	798	343	ABC transporter, ATP-binding protein	1
contig00732	1256	1399	ABC transporter, ATP-binding protein	1
contig00819	1514	1296	ABC transporter, permease protein	
contig00568	111	659	ABC transporter, permease protein	

contig00819	1448	1104	ABC-type anion transport system, duplicated permease component	
contig00568	267	533	ABC-type anion transport system, duplicated permease component	
contig00084	696	280	ABC-type multidrug transport system, ATPase component	
contig00038	696	181	ABC-type multidrug transport system, ATPase component	
contig00893	1702	1806	ATP synthase gamma chain (EC 3.6.3.14)	
contig01037	60	473	ATP synthase gamma chain (EC 3.6.3.14)	
contig00940	153	620	ATP-dependent Clp protease proteolytic subunit (EC 3.4.21.92)	
contig00021	143	3	ATP-dependent Clp protease proteolytic subunit (EC 3.4.21.92)	
contig00188	781	191	AmpG permease	
contig00364	329	871	AmpG permease	
contig00710	219	626	Aspartokinase (EC 2.7.2.4)	
contig00760	2878	2747	Aspartokinase (EC 2.7.2.4)	
contig00455	1584	1123	Aspartokinase (EC 2.7.2.4) / Homoserine dehydrogenase (EC 1.1.1.3)	
contig00710	147	1088	Aspartokinase (EC 2.7.2.4) / Homoserine dehydrogenase (EC 1.1.1.3)	
contig00622	1635	753	Cell division protein FtsI [Peptidoglycan synthetase] (EC 2.4.1.129)	
contig00260	1783	1143	Cell division protein FtsI [Peptidoglycan synthetase] (EC 2.4.1.129)	
contig00349	1691	453	Cell division protein FtsI [Peptidoglycan synthetase] (EC 2.4.1.129)	

contig00438	776	15	Chaperone protein HtpG	
contig00169	40	474	Chaperone protein HtpG	
contig01186	414	542	Chemotaxis protein CheV (EC 2.7.3.-)	
contig00174	274	1044	Chemotaxis protein CheV (EC 2.7.3.-)	
contig00107	1318	1235	Chromosome (plasmid) partitioning protein ParA / Sporulation initiation inhibitor protein Soj	
contig00555	141	34	Chromosome (plasmid) partitioning protein ParA / Sporulation initiation inhibitor protein Soj	
contig00573	705	57	Cytochrome O ubiquinol oxidase subunit I (EC 1.10.3.-)	
contig01024	1035	1610	Cytochrome O ubiquinol oxidase subunit I (EC 1.10.3.-)	
contig00066	50	922	Cytochrome d ubiquinol oxidase subunit I (EC 1.10.3.-)	
contig00918	1474	335	Cytochrome d ubiquinol oxidase subunit I (EC 1.10.3.-)	
contig00066	2055	2639	Cytochrome d ubiquinol oxidase subunit II (EC 1.10.3.-)	
contig00918	202	22	Cytochrome d ubiquinol oxidase subunit II (EC 1.10.3.-)	
contig00361	988	293	D-alanyl-D-alanine carboxypeptidase (EC 3.4.16.4)	
contig00230	81	1031	D-alanyl-D-alanine carboxypeptidase (EC 3.4.16.4)	
contig00233	1037	327	D-alanyl-D-alanine carboxypeptidase (EC 3.4.16.4)	
contig01023	582	944	DNA repair protein RecN	
contig00769	6655	7104	DNA repair protein RecN	
contig00711	194	829	Fatty acid desaturase (EC 1.14.19.1); Delta-9 fatty acid desaturase (EC 1.14.19.1)	

contig00812	5418	5784	Fatty acid desaturase (EC 1.14.19.1); Delta-9 fatty acid desaturase (EC 1.14.19.1)	
contig00065	396	283	Flagellar biosynthesis protein FlhF	
contig00626	260	1030	Flagellar biosynthesis protein FlhF	
contig00641	1369	917	Flagellar synthesis regulator FleN	
contig00626	1152	1424	Flagellar synthesis regulator FleN	
contig00029	64	663	Gamma-glutamyltranspeptidase (EC 2.3.2.2)	
contig00683	601	208	Gamma-glutamyltranspeptidase (EC 2.3.2.2)	
contig00699	15	593	Glutathione synthetase (EC 6.3.2.3)	
contig00050	971	360	Glutathione synthetase (EC 6.3.2.3)	
contig00856	2851	2928	Glycerophosphoryl diester phosphodiesterase (EC 3.1.4.46)	
contig00965	2672	2830	Glycerophosphoryl diester phosphodiesterase (EC 3.1.4.46)	
contig00535	447	572	Homogentisate 1,2-dioxygenase (EC 1.13.11.5)	
contig00219	37	558	Homogentisate 1,2-dioxygenase (EC 1.13.11.5)	
contig00100	1267	1169	Hypothetical protein YggS, proline synthase co-transcribed bacterial homolog PROSC	
contig00660	263	580	Hypothetical protein YggS, proline synthase co-transcribed bacterial homolog PROSC	
contig00414	140	69	IcmB (DotO) protein	
contig00394	726	493	IcmB (DotO) protein	
contig00446	4763	4647	Integral membrane protein YggT, involved in response to	

			extracytoplasmic stress (osmotic shock)	
contig00617	745	864	Integral membrane protein YggT, involved in response to extracytoplasmic stress (osmotic shock)	
contig01008	128	559	Integrase, catalytic region	
contig00116	1800	1753	Integrase, catalytic region	
contig00514	2056	117	Lead, cadmium, zinc and mercury transporting ATPase (EC 3.6.3.3) (EC 3.6.3.5); Copper-translocating P-type ATPase (EC 3.6.3.4)	
contig00619	625	215	Lead, cadmium, zinc and mercury transporting ATPase (EC 3.6.3.3) (EC 3.6.3.5); Copper-translocating P-type ATPase (EC 3.6.3.4)	
contig00298	3	1424	Long-chain-fatty-acid--CoA ligase (EC 6.2.1.3)	
contig00191	1	1021	Long-chain-fatty-acid--CoA ligase (EC 6.2.1.3)	
contig00717	292	1212	Mannose-1-phosphate guanylyltransferase (GDP) (EC 2.7.7.22)	
contig00087	444	136	Mannose-1-phosphate guanylyltransferase (GDP) (EC 2.7.7.22)	
contig00821	4	87	Membrane alanine aminopeptidase N (EC 3.4.11.2)	
contig00401	2537	197	Membrane alanine aminopeptidase N (EC 3.4.11.2)	
contig00579	206	502	Methyl-accepting chemotaxis protein	
contig00074	404	39	Methyl-accepting chemotaxis protein	
contig01014	1364	732	Methyl-accepting chemotaxis protein	
contig01046	597	526	Methyl-accepting chemotaxis protein	
contig00109	2170	1661	N-acetylglucosamine-1-phosphate uridyltransferase (EC 2.7.7.23) / Glucosamine-1-phosphate N-acetyltransferase (EC 2.3.1.157)	
contig01037	2463	2924	N-acetylglucosamine-1-phosphate uridyltransferase (EC 2.7.7.23) / Glucosamine-1-phosphate N-acetyltransferase (EC 2.3.1.157)	

contig00840	19	264	N-acetylmuramoyl-L-alanine amidase (EC 3.5.1.28)	
contig00068	532	777	N-acetylmuramoyl-L-alanine amidase (EC 3.5.1.28)	
contig00577	593	45	N-acetylmuramoyl-L-alanine amidase (EC 3.5.1.28)	
contig00603	1148	255	NAD-specific glutamate dehydrogenase (EC 1.4.1.2), large form	
contig00564	231	3362	NAD-specific glutamate dehydrogenase (EC 1.4.1.2), large form	
contig01053	1246	398	Nucleoside permease NupC	
contig00096	113	1171	Nucleoside permease NupC	
contig00040	381	977	Oligopeptide transport ATP-binding protein OppD (TC 3.A.1.5.1)	
contig00303	7495	6725	Oligopeptide transport ATP-binding protein OppD (TC 3.A.1.5.1)	
contig00195	215	310	ParA family protein	
contig00756	3859	4146	ParA family protein	
contig00151	743	12	Phage portal protein	
contig00041	68	202	Phage portal protein	
contig00119	2057	2521	Phage portal protein	
contig00114	1313	516	Phage terminase large subunit	
contig00119	282	965	Phage terminase large subunit	
contig00142	292	699	Potassium uptake protein TrkH	
contig00971	533	18	Potassium uptake protein TrkH	
contig00160	4222	4419	RND multidrug efflux transporter; Acriflavin resistance protein	2

contig00633	47	447	RND multidrug efflux transporter; Acriflavin resistance protein	2
contig00474	46	372	RND multidrug efflux transporter; Acriflavin resistance protein	2
contig00830	960	640	RND multidrug efflux transporter; Acriflavin resistance protein	2
contig00023	189	614	RND multidrug efflux transporter; Acriflavin resistance protein	2
contig00390	2794	3009	RND multidrug efflux transporter; Acriflavin resistance protein	2
contig00850	694	996	Single-stranded DNA-binding protein	
contig00848	715	413	Single-stranded DNA-binding protein	
contig00744	559	870	Single-stranded DNA-binding protein	
contig00266	12498	13463	Sulfate permease	
contig00110	355	2	Sulfate permease	
contig00593	494	120	Sulfate permease	
contig00567	736	140	Transcription-repair coupling factor	
contig00571	361	2196	Transcription-repair coupling factor	
contig01155	268	621	Transposase	3
contig00989	538	107	Transposase	3
contig00667	1192	1124	Transposase	3
contig00366	792	607	Transposase	3
contig00036	152	469	Transposase	3
contig00545	129	748	Transposase	3
contig00589	76	417	Transposase	3
contig00302	312	593	Transposase	3
contig00491	279	67	Transposase	3
contig00773	914	196	Transposase	3

contig00158	3614	4173	Tyrosine-specific transport protein	
contig00070	6252	6725	Tyrosine-specific transport protein	
contig00167	8955	8905	UDP-N-acetylmuramoylalanyl-D-glutamate--2,6-diaminopimelate ligase (EC 6.3.2.13)	
contig00622	322	23	UDP-N-acetylmuramoylalanyl-D-glutamate--2,6-diaminopimelate ligase (EC 6.3.2.13)	
contig00459	860	180	UDP-glucose dehydrogenase (EC 1.1.1.22)	
contig00140	2619	2924	UDP-glucose dehydrogenase (EC 1.1.1.22)	
contig01028	902	192	UTP--glucose-1-phosphate uridylyltransferase (EC 2.7.7.9)	
contig00459	96	3	UTP--glucose-1-phosphate uridylyltransferase (EC 2.7.7.9)	
contig00476	14	646	UTP--glucose-1-phosphate uridylyltransferase (EC 2.7.7.9)	
contig00874	556	74	Xaa-Pro aminopeptidase (EC 3.4.11.9)	
contig00580	10992	11645	Xaa-Pro aminopeptidase (EC 3.4.11.9)	
contig00173	1473	241	amino acid permease family protein	
contig00874	3074	3355	amino acid permease family protein	
contig00510	163	3	metalloprotease, insulinase family	
contig00067	2633	615	metalloprotease, insulinase family	
contig00240	2999	3946	tRNA uridine 5-carboxymethylaminomethyl modification enzyme GidA	
contig00555	1572	922	tRNA uridine 5-carboxymethylaminomethyl modification enzyme GidA	

Notes: I don't think that 1, 2, or 3 are worth worrying about. They are clearly cases where we have generic annotations for proteins, and have not resolved them.

This covers 125 contigs, about 1/8th of the total.

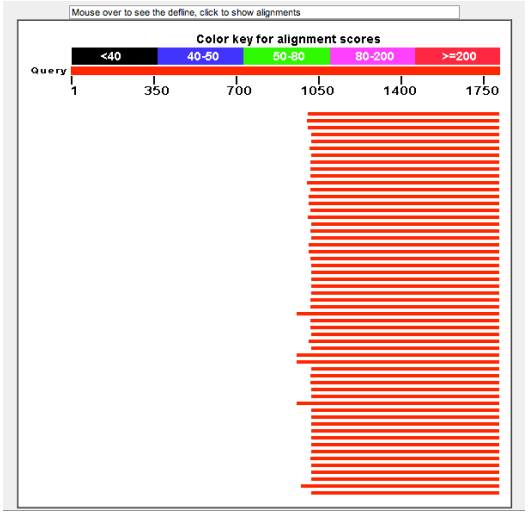
Example 1:

I took one of these pairs, the first one on the list:

contig00973	88	243	4-hydroxyphenylpyruvate dioxygenase (EC 1.13.11.27)
contig00423	1134	1799	4-hydroxyphenylpyruvate dioxygenase (EC 1.13.11.27)

Extract the sequences from the fasta file, and compare them to the non-redundant database using BLASTX. **Note:** because I am using BLASTX the query sequence (the contigs) coordinates will be in bp, while the database sequence (the protein) coordinates will be in amino acids.

Contig00423 is clearly the 4-hydroxyphenylpyruvate dioxygenase protein, and is the start of the protein at the end of the contig. See the blast summary figure.



Clearly the gene starts at position 1050 and runs off the end of the contig.

This is the alignment of the best hit. Note that the subject here starts at the 2nd amino acid and proceeds through 279 of 365 amino acids.

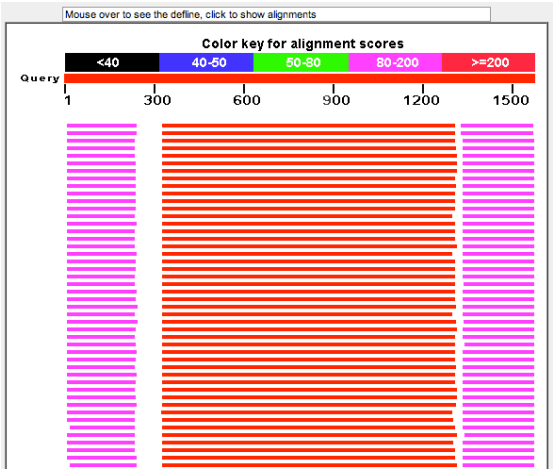
```
>ref|YP_432268.1|  4-hydroxyphenylpyruvate dioxygenase [Hahella chejuensis KCTC 2396]
 gb|ABC27843.1|  4-hydroxyphenylpyruvate dioxygenase [Hahella chejuensis KCTC 2396]
Length=365

 GENE ID: 3839181 hppD | 4-hydroxyphenylpyruvate dioxygenase
[Hahella chejuensis KCTC 2396] (10 or fewer PubMed links)
```

```
 Score =  308 bits (788),  Expect = 2e-81
 Identities = 159/278 (58%), Positives = 194/278 (70%), Gaps = 9/278 (3%)
 Frame = +3

Query  1005  SSTTINPMQTQGFAFLEFTA--SKDSDHFTKTFTMLGFSKIATHLSKDVELWQQGDIHFF  1178
              S+   NP+ T GF F+EF A     ++     F  LGF+ IA H  K + L++QGDI+F
Sbjct  2     SNVYTNPLGTDGFEFVEFGAPDKAGAESLKPLFETLGFTAIARHKRKAITLYRQGDINFL  61

Query  1179  VNRDVNSVSQQFSNQHGPCTSAMGFRVKDADFAFERALTLGAEAFSGEGV----INVPAI  1346
              VN +V S   +FS  HGPC  AM FRVKDA  AFE A++ GA+ F    V    + +PA+
Sbjct  62    VNENVPSYFHKFSMDHGPCACAMAFRVKDAQKAFEYAVSQGAKPFDQGDVQGEEVRIPAV  121

Query  1347  YGIGGSILYFVS---EDYNLENEFEIIAGVDAASASQGLVNLDHVTHNVLRGNMDKWAGF  1517
              YGIGGS+LYFV    ED   + +FE   GVD      GL  LDH+THNV RGNMD WAGF
Sbjct  122   YGIGGSVLYFVDKYGEDSIYDIDFEYFPGVDKHPKGLGLTTLDHLTHNVFRGNMDTWAGF  181

Query  1518  YEKLFNFREIRYFDIEGKLTGLKSRAMTSPCGQIRIPINESSDDKSQIEEFIRAFNGEGI  1697
              YE++ NFREIRYFDIEGKLTGL SRAMTSPC +IRIPINES+DDKSQIEE+++ + GEGI
Sbjct  182   YERIGNFREIRYFDIEGKLTGLHSRAMTSPCCKIRIPINESADDKSQIEEYLQEYKGEGI  241

Query  1698  QHIALTSNDIYKTVENIRTNGVQFLDTPDTYYEGISKR  1811
              QHIAL++ DIYKTV ++R  GV F+DTPDTYYE I  R
Sbjct  242   QHIALSTEDIYKTVRDMREAGVVFMDTPDTYYEKIDTR  279
```

Notice that this sequence extends to position 1811 on this contig, and the contig is 1811 bp long, thus this is exactly the end.

The first sequence, however, is more complex because the blast hits with default values are dominated by another protein on that contig.

To visualize the correct alignment, you need to increase the number of “max target sequences” under Algorithm parameters (before you submit the BLAST) (I used 500 for this) and then change the “graphical overview” to 500 and “alignments” to 500. Then you will see a picture like this:



Looking at the alignment we see that the start is at bp 13 on the contig, and extends through bp 246, and this covers amino acids 286 through 363.

```
>ref|ZP_02189132.1|  4-hydroxyphenylpyruvate dioxygenase
[alpha proteobacterium BAL199]
 gb|EDP64091.1|  4-hydroxyphenylpyruvate dioxygenase [alpha
proteobacterium BAL199]
Length=363

 Score =  118 bits (295),  Expect = 3e-24
 Identities = 56/78 (72%), Positives = 66/78 (85%), Gaps =
0/78 (0%)
 Frame = +1

Query  13
VPWHNESIKDLHENKILIDGAPTAEGGLLLQIFTDTVIGPAFFEIIQRKGDEGFGEGNF
Q  192
            +P H E++  L E  IL+DGAPT +GG LLQIFT TVIGP
FFE+IQRKGDEGFGEGNF+
Sbjct  286
LPGHGENVAALQERGILMDGAPTEDGGRLLQIFTKTVIGPIFFELIQRKGDEGFGEGNF
R  345

Query  193  ALFDSIELDQVRRGVLKA  246
            ALF+SIE DQ++RGVLKA
Sbjct  346  ALFESIEQDQIKRGVLKA  363
```

The gap here is from 279 on contig00423 through 286 at the start of this contig: 9 amino acids or 27 bp. However, accommodating the 13 bp at the start of this alignment, we have a gap of 14bp between the two contigs.

Thus, we conclude the orientation should be:

Contig00423 and then a gap of 14 bp and then contig00973 and both contigs are in the correct orientation.

Example 2.

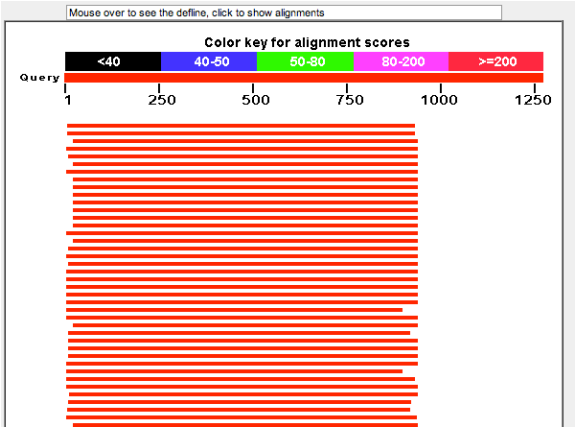
As a second example, I took

contig00188	781	191	AmpG permease
contig00364	329	871	AmpG permease

These appear to be adjacent, but the reading frames are on different strands.

Repeating the BLAST at NCBI, I found:

The smaller contig, contig00188 (1274 bp) has a protein at its left end:



and this is the AmpG protein.

```
>ref|NP_819241.1|  AmpG [Coxiella burnetii RSA 493]
 ref|ZP_01946587.1|  AmpG protein [Coxiella burnetii 'MSU
Goat Q177']
 ref|YP_001596149.1|  AmpG protein [Coxiella burnetii RSA
331]
 9 more sequence titles
 Length=433

 GENE ID: 1208071 ampG | AmpG [Coxiella burnetii RSA 493]
(10 or fewer PubMed links)

 Score =  318 bits (816),  Expect = 8e-85
 Identities = 157/308 (51%), Positives = 215/308 (70%),
Gaps = 0/308 (0%)
 Frame = -2

Query  934
LSLNRRTTAVFLLTFSSSLPLILVGSTLSAWYTVAGVSLLGIGALSMVQQPYIFKFLWA
P  755
            L  NRR   V  L+FSS+LPL+L+  TL AWYT AGVSL+
IG+LS++Q  Y+ KF+WAP
Sbjct  12
LLFNRRVMTVLFLSFSSTLPLVLLTGTLQAWYTAAGVSLMVIGSLSLLQYAYLVKFVWA
P  71

Query  754
LMDRFSLFGWCRRRSWILLTQVLLVIGLVLMAYTGPKQYPFGLAMIALIFSFFSASQDI
A  575
```

```
             MDR++  G  RRRSWILLTQ+ L   L +MA+  P  +P+ L  ++
+ +F SASQD+
Sbjct  72
FMDRYAPLGLGRRRSWILLTQISLAGALAVMAFMNPAYHPWELFWLSAVVAFISASQDV
V  131

Query  574
IDAYRIDVLHANELGVGAAITSLAGRIAMLFGGAVALVFAAEIGWQNTYLIMAAVMALE
I  395
            ID YR+DVL   E G G A+T++ GR+AML GG +AL+ AA+ GW+
TYL MA +M ++I
Sbjct  132
IDGYRVDVLRQKERGFGVAVTTVGGRMAMLVGGGLALILAADYGWRATYLTMALLMLIQ
I  191

Query  394
LATFAAPALRNEEATPRSLQESIVGPLKDFVRREHALSILIFILTYKMCDALALALNTP
F  215
            + T  AP        P +L  ++V P ++ + R+H + IL+ +L YK
DALALALNT F
Sbjct  192
IPTLTAPNPEKPVYPPVTLHAAVVAPFRELLTRKHIVMILLLVLIYKFGDALALALNTT
F  251

Query  214
LIRGVGFSLLEMGTIYKTMSLVASLLGTVVGGIFYKRLGLYRTLMCYGILQALSNLAYM
G  35
            L+RGVGF+LLE+G  YK  S++ASLLG   GG F  RLGLYR+LM
+G LQ ++NL++M
Sbjct  252
LMRGVGFTLLELGYSYKLTSVIASLLGGFAGGFFMPRLGLYRSLMVFGFLQTVTNLSFM
V  311

Query  34   LALASKSY  11
            LAL  K++
Sbjct  312  LALVGKNF  319
```

From the alignment we notice that the start is at position 934 bp on the contig, and extends to position 11 bp. Thus the ORF is on the negative strand. In the database sequence, the alignment extends from amino acids 12 through 319 of 433 amino acids in the sequence.

```
>ref|YP_001425223.1|  AmpG [Coxiella burnetii Dugway 5J108-
111]
 gb|ABS77532.1|  AmpG [Coxiella burnetii Dugway 5J108-111]
Length=433

 GENE ID: 5458939 ampG | AmpG [Coxiella burnetii Dugway
5J108-111]
```

```
(10 or fewer PubMed links)

 Score =  254 bits (650),  Expect = 4e-65
 Identities = 177/408 (44%), Positives = 254/408 (63%),
Gaps = 10/408 (2%)
 Frame = +2

Query  302
LTKRRILTVFLLGFSSGLPFALIAGTLQAWFTTAGLDLKTIGAASLIGLPYTLKPFIAP
I  481
             L  RR++TV  L FSS LP  L+ GTLQAW+T AG+ L  IG+
SL+   Y +K   AP
Sbjct  13
LFNRRVMTVLFLSFSSTLPLVLLTGTLQAWYTAAGVSLMVIGSLSLLQYAYLVKFVWAP
F  72

Query  482
IDryrlpllgrrrgwlllfqfalliailfMSTLKPTHSVLVLGQTVPLLMIIGVCVAFF
S  661
             +DRY    LGRRR W+LL Q +L   +  M+ + P +    L
+   VAF S
Sbjct  73    MDRYAPLGLGRRRSWILLTQISLAGVLAVMAFMNPAYHPWEL----
---FWLSAVVAFIS  125

Query  662
TCQDIVINAYQTEVLYENERGLGASLGVTGWRISFILSGSIALILASHLGWHLTYIVMS
L  841
               QD+VI+ Y+ +VL + ERG G ++   G R++ ++ G
+ALILA+  GW  TY+ M+L
Sbjct  126
ASQDVVIDGYRVDVLRQKERGFGVAVTTVGGRMAMLVGGGLALILAADYGWRATYLTMA
L  185

Query  842
CMLIGVFATLFGPNPDNEGQPPMTFVKAIYEPFVDFFSRQKLPTAllvillilTYKIGD
A  1021
              MLI +  TL  PNP+    PP+T   A+  PF +  +R+ +
++++LL+L YK GDA
Sbjct  186   LMLIQIIPTLTAPNPEKPVYPPVTLHAAVVAPFRELLTRKHI---
VMILLLVLIYKFGDA  242

Query  1022
LALALNTTFLIKGVGFSLDaigvinknigiigsiaggiiggllmlrlslfrslmlfglV
Q  1201
             LALALNTTFL++GVGF+L  +G   K   +I S+ GG  GG  M
RL L+RSLM+FG +Q
Sbjct  243
LALALNTTFLMRGVGFTLLELGYSYKLTSVIASLLGGFAGGFFMPRLGLYRSLMVFGFL
Q  302
```

```
Query  1202
AIANLGYMWLAIVGKNYSVFVITAFSEHFFSGMGTAAFVALLMSQCNLNYTATQYALFT
S  1381
             + NL +M LA+VGKN+ V + + FS++F  G+ TAAFVA L++ C
Y+ATQYA+ ++
Sbjct  303
TVTNLSFMVLALVGKNFPVMIGSIFSDYFCGGLSTAAFVAFLIALCKREYSATQYAILS
A  362

Query  1382
LSLFGRTWLGPIAASMVDHFGWAQFFFWSFIVSLPCLFFVWLLRHKLE  1525
             L    R  +GP+AA +V H GWA F+  +F    P L  +  LR
+++
Sbjct  363
LMAVSRVVIGPVAAYLVKHMGWAPFYLITFFAGFPALLILRWLRDRID  410
```

[Note this wasn't the best AmpG hit, but it is the same protein as the one above]

In this case, the query sequence runs from 302 bp through 1525 bp on the contig, and amino cids 13 through 410 of 433 on the database sequence.

Therefore, these sequences should overlap! The first contig matched amino acids 12 → 319 while the second sequence matched 13 → 410. Yikes. Recall that the first sequence was complementary to the amino acid sequence, so I decided to look at the alignment by a MSA.

This shows that the sequences overlap, but really should not be joined together. Here is a pertinent piece of the alignment:

```
contig00364
ATGCATGCTGATTGGTGTTTTTGCCACTCTTTTTGGACCTAACCCCGACAATGAAGGTC
A
contig00188      ----------------------------------------
-----GCCATTAAAAATA

* ** **    *

contig00364
ACCCCCAATGACTTTTGTCAAAGCTATTTATGAGCCTTTTGTTGACTTTTTCTCCAGGC
A
contig00188
ACTTTTACTGGCCAGAGCCAGTCCCATATACGCTAAATTAGAAAGCGCCTGTAATATTC
C
                 **    * ** *    * **   * ** ** *     ** *
*   *     *  *
```

```
contig00364
AAAACTACCTACCGCCCTGCTCGTTATTTTACTGATTTTAACCTATAAAATCGGTGATG
C
contig00188      ATAACACATCAGCGTAC-
GATAAAGACCTAATCGCTTATAAAAAATCCCACCAACGACGG
                 * ***     * **  * * *    *  * *  * ** ***
**   * *   ** *

contig00364      ATT--GGCACTGGCACT-
CAATACCACCTTTTTAATTAAAGGGGTTGGCTTTTCACTGGA
contig00188
TTCCCAGCAGTGAAGCTACCAAACTCATTGTCTTATAAATCGTGCCCATTTCTAGCAAT
G
                  *    *** **   ** * * **    * * * ** **  *
*     ** *  *

contig00364
TGCGATTGGCGTTATTAATAAAAATATTGGCATTATCGGCTCAATTGCCGGCGGCATTA
T
contig00188      AAAACCCAACCCCTCTAATTAAAAAAGGTGTATTA--
AGTGCCAATGCCAA-AGCATCAC
                           *     **** **** *   * ****   *  *
* ****    **** *

contig00364
CGGTGGTTTATTGATGCTAAGGCTCTCTTTATTTCGCTCATTAATGCTATTTGGACTTG
T
contig00188      --ACATTTTAT--ATGTTAAAAT------------
GAAAATCAATATTGAAAGCGCATGC
                         *****  *** ***              *   **
***  *    *  * **

contig00364
TCAAGCCATCGCTAACTTAGGTTATATGTGGCTAGCTATTGTTGGCAAAAATTATAGTG
T
contig00188      TCTCGCCG-CACAAAATC---
TTTCAGTGGACCGACAATGGATTCTTGTAATGATCGTGG
                 **  ***  * * ** *    **  *   * *   * ** *
*      *** ** ***

contig00364
TTTCGTGATCACAGCCTTTTCTGAGCACTTTTTTAGTGGCATGGGAACTGCTGCCTTTG
T
contig00188
AGTCGCTTCCTCATTCCTTAGTGCCGGCGCTGCAAATGTTGCTAAAATTTCTAAGGCCA
T
                   ***    * **  * **  **    *  *   * **
** * **       *
```

```
contig00364
CGCTTTATTAATGTCACAATGCAATTTAAACTATACCGCAACCCAATATGCCTTATTTA
C
contig00188      CACCGC-----TGCCATAATTAAAT---
ACGTATTCTGCCAACCTATCTCTGCTGCAAAA
                 * *        ** ** ***  ***   *  *** * ** *
** ** *    *    *

contig00364
CTCACTCTCATTATTTGGCCGCACCTGGCTCGGCCCTATTGCCGCCAGCATGGTTGATC
A
contig00188      ACTAAGGCAACAGC--GCCCCCAAATAGCATCGC---
AATACGGCCAGCTAAACTGGTAA
                    *     *      * ** **  * **   **   * * *
******     ** * *

contig00364
CTTTGGCTGGGCGCAATTTTTCTTCTGGAGCTTTATTGTTTCGCTGCCGTGCTTGTTTT
T
contig00188      TTGCAGCCCC-CACACCTAATTCATTGGCA-
TGCAAAACATCAATACGATAGGCATCAAT
                  *   **    * **  *  *    ***   *  *     **
* *  *     *   *

contig00364
TGTCTGGCTGTTACGTCATAAACTAGAAAAAGATCATCAAGATAGAATCAATAAACAAG
G
contig00188      TG-
CAATATCTTGAGAAGCAGAGAAAAAAGAAAAAATCAAAGCAATCATAGCCAAGCCAA
                 ** *    * **  *    * *  * *** * *  *****
*     *   **

contig00364
ACAGGAAGTCGCCTGTTCCTAAAATTAATCAATTAACTTAAAAATTGCTTCGCCATATG
C
contig00188      ATGGGTACTGCTTGGGGCCCGTA--
TACGCCATAAGCACCAAGCCAA--TCACCAACAGA
*  ** * *     *  **   *  **  * ** * *   **       ** ***   *
```

So, my next suspicion is that there are two paralogs of this protein in the genome. Lets take a look at Coxiella (the example above):

If we start here: http://www.ncbi.nlm.nih.gov/genomes/lproks.cgi and go down to Coxiella, there is a link to the genome blast: http://www.ncbi.nlm.nih.gov/genomes/geblast.cgi?gi=23256 from there, we can paste in the RefSeq ID from the first alignment above, NP_819241.1 and BLAST against the proteins in one of those genomes. Hmm, that reveals one per Coxiella genome, so that is not our explanation.

A possible explanation is that both these hits are weak homologs – 44% and 51% identical in the BLAST hits, and so maybe it is just chance?

I don't yet know the explanation for this, but clearly we should not join contigs when BLAST suggests there is an overlap between the ORFs on both contigs.

Example 3

The third example is

contig00459	860	180	UDP-glucose dehydrogenase (EC 1.1.1.22)
contig00140	2619	2924	UDP-glucose dehydrogenase (EC 1.1.1.22)

Which has the same feature as above … the alignments are on different strands. Lets start with our BLASTs, just as before.

Contig 00459

```
>ref|ZP_06640176.1|  UDP-glucose 6-dehydrogenase [Serratia
odorifera DSM 4582]
 gb|EFE94641.1|  UDP-glucose 6-dehydrogenase [Serratia
odorifera DSM 4582]
Length=447

 Score =  299 bits (766),  Expect = 3e-79
 Identities = 163/243 (68%), Positives = 193/243 (80%),
Gaps = 0/243 (0%)
 Frame = -1

Query  863
LVVMDVRSAELTKYAANAMLATKISFMNEMSQLAERVGADIEMVRQGIGSDSRIGYHFI
Y  684
            +++MD+RSAELTKYAAN MLATKISFMNE+S LAE +GADIE
VRQGIGSDSRIGYHFIY
Sbjct  198
MILMDIRSAELTKYAANCMLATKISFMNEISNLAELLGADIEKVRQGIGSDSRIGYHFI
Y  257

Query  683
PGCGYGGSCFPKDVRALVHTAAEHGFDTKILGAVqevnekqkelllekvLREFQGDIQG
K  504
            PGCGYGGSCFPKDV+AL+ T+   G+  K+L AV++VN +QK  L
+   F  D+ GK
Sbjct  258
PGCGYGGSCFPKDVQALIRTSEHIGYQPKLLQAVEQVNYQQKYKLTTFIKHYFGEDLAG
K  317
```

```
Query  503
TFALWGLAFKPKTDDIREAPSRVLMEGLWQHGAKVQAYDPAAMDNIQAVYGARddlyla
t  324
            TFALWGLAFKP TDD+REA +RVLME LW+ GAKVQAYDP AM+  Q
+YG RDDL L
Sbjct  318
TFALWGLAFKPNTDDMREASARVLMETLWEAGAKVQAYDPEAMNEAQRIYGHRDDLKLM
G  377

Query  323
sassaltgadaLIVVTEWTEFRSPDFNVIKKTLNQPLVIDGRNIYDAEHLESLGITYRC
I  144
            +  +AL GADAL++ TEW  FR+PDF+VIK TL QP++ DGRN+YD
E LES G TY  I
Sbjct  378
TKEAALQGADALVICTEWQNFRAPDFDVIKSTLKQPVIFDGRNLYDPERLESRGFTYYA
I  437

Query  143  GRG  135
            GRG
Sbjct  438  GRG  440
```

Summarizing:

DNA	Start: 863	End: 135	Contig length: 929 bp
Protein	Start: 198	End: 440	Protein length: 447 aa

Contig 00140

This is a more extreme case of the number of good hits. This contig has >500 hits to a glucose dehydrogenase. Expanding number of hits/alignments to 1,000 showed:

```
>ref|ZP_06640176.1|  UDP-glucose 6-dehydrogenase [Serratia
odorifera DSM 4582]
 gb|EFE94641.1|  UDP-glucose 6-dehydrogenase [Serratia
odorifera DSM 4582]
Length=447

 Score =  164 bits (414),  Expect = 1e-37
 Identities = 80/135 (60%), Positives = 104/135 (78%), Gaps
= 0/135 (0%)
 Frame = +3

Query  2607
MRVTVFGAGYVGLVTAACFADLGNQVICVDVDEKKLAQLAEGKSPIYEPGLDELLLRGQ
E  2786
             M+VTVFG GYVGLV AA  A++G+ V+C+DVDE+K+  L +G
PI+EPGL  L+ +  E
```

```
Sbjct  1
MKVTVFGIGYVGLVQAAVLAEVGHDVMCIDVDERKVENLKKGNIPIFEPGLTPLVQQNF
E  60

Query  2787
SGNLEFTADIQSAVEQGEFIFIAVGTPSEESGSADLQYVLAVAKSIGEYMNGYKLVIDK
S  2966
            +G L FT D Q+ V  G   FIAVGTP +E GSADL+YV AVA++I
E+M  +K+VIDKS
Sbjct  61
AGRLHFTTDAQAGVAHGTIQFIAVGTPPDEDGSADLKYVTAVARTIAEHMTDHKVVIDK
S  120

Query  2967  TVPLGTADKVRQVIS  3011
              TVP+GTADKVRQV++
Sbjct  121   TVPVGTADKVRQVMT  135
```

Summarizing:

DNA	Start: 2607	End: 3011	Contig length: 3018 bp
Protein	Start: 1	End: 135	Protein length: 447 aa

[Again, note that I didn't take the best hit, but the same protein as above]

This shows that the protein starts on Contig 00140, and ends at position 135 aa. There are an additional 7 bp on this sequence. Then, it continues at position 863 of the negative strand of Contig 00459 (66 bp into the sequence), and ends at position 135.

The gap in the protein is from aa 135 to 198, 63 amino acids, 189 bp. The extraneous sequence is 73bp, so the overall gap is 122 bp.

The right end of Contig 00140 joins to the reverse complement of Contig 00459 and the gap is 122 bp.

Example 4.

contig00710	219	626	Aspartokinase (EC 2.7.2.4)
contig00760	2878	2747	Aspartokinase (EC 2.7.2.4)

```
Contig 00710
>ref|ZP_05109721.1|  bifunctional aspartokinase I/homeserine dehydrogenase I [Legionella
drancourtii LLAP12]
 gb|EET12579.1|  bifunctional aspartokinase I/homeserine dehydrogenase I [Legionella
drancourtii LLAP12]
Length=813

 Score =  367 bits (943),  Expect = 1e-99
 Identities = 197/407 (49%), Positives = 274/407 (68%), Gaps = 8/407 (1%)
```

```
 Frame = +3

Query  3     VDASQVLILADNE----VDWEKSAENL-AKLNLSDYDQVVITGFIAGNDQRVQLTLGRNG  167
             VDAS +L + +      +DW+K+   L A L+   +DQ++ITGFIA      +  LGRNG
Sbjct  145   VDASTILFIFEKNGIICIDWQKTQRALNAFLHDKVFDQIIITGFIASTLDGKRTILGRNG  204

Query  168   SDYSAAIFAKILGTDKLIIWTDVDGIMSADPRKVPSAFVLPCISYQEALELAYFGATVLH  347
              D+SAAIFAK+L    L IWTDVDG+ +ADP KV SAFV+  +SYQEA ELAYFGA VLH
Sbjct  205   GDFSAAIFAKLLRAKSLTIWTDVDGVYTADPNKVRSAFVIEELSYQEASELAYFGAKVLH  264

Query  348   PSTIEPMMSVGSTIFIKNSFKPDEPGTIIGQSNEPPSSLIKGITCVESAALLNIEGSGMA  527
             P TI P   +   I IKNSF P   GT I  ++      IKG+T +++ AL+NIEG+G+
Sbjct  265   PMTIAPAFELKIPIIIKNSFNPQAKGTYITGTS---IKSIKGLTSIDNVALINIEGAGIL  321

Query  528   GVPGSAERIFQILRQGHISVILISQASSEQSICVAIKSDQAMRARQLLTQHFRYEIEYGK  707
             GV G A R+FQ L Q +ISVILI+QASSE SIC AI ++QA  A   L +HF++E+E+ +
Sbjct  322   GVSGVASRVFQTLHQKNISVILIAQASSEYSICFAIANEQADNAMNALEEHFQFELEHQQ  381

Query  708   IKSIEADESCSIIAAVGDGMVGQRGIAAKICHALAMANVNIKAIAQGVAERNISLVIQRY  887
              + I  D+SC I++AVGDGM+G  G +AK+  +LA AN+NI+A++QG +ERNIS+VI
Sbjct  382   FQRITMDKSCGILSAVGDGMIGAIGGSAKLFSSLAKANINIRAVSQGSSERNISVVINSS  441

Query  888   EAQKAMRAIHSGLYLSHKTISVGLIGPGRIGATLLSQLKQQQDILKKEHGLALRLRGVAN  1067
             +  KA++A H+  YLS +TIS+GLIGPG +G+ LLSQ+    + LK      L +RG+ N
Sbjct  442   DMNKALQAAHAEFYLSRETISIGLIGPGHVGSCLLSQIHDALERLKISSQANLLVRGIMN  501

Query  1068  SKQMLLDEHEIDLGSWQDEFGVKGAEIDLNVFIDHVVSDFIPHSLII  1208
             S+ MLL    I+L +W+++      + +L  FI H++ D IPH+++I
Sbjct  502   SRTMLLSHCSINLSTWREQLSQCEVKANLQGFIKHILVDDIPHAVLI  548
```

Contig 00760

```
>ref|YP_003921249.1|  aspartokinase II alpha subunit (aa 1->408) and beta subunit
(aa 246->408) [Bacillus amyloliquefaciens DSM7]
 emb|CBI43779.1|  aspartokinase II alpha subunit (aa 1->408) and beta subunit
(aa 246->408) [Bacillus amyloliquefaciens DSM 7]
Length=409

Score =  222 bits (566),  Expect(2) = 1e-65
 Identities = 140/327 (43%), Positives = 202/327 (62%), Gaps = 8/327 (2%)
 Frame = -3

Query  2878  VIVQKYGGTSVESIEKIHKIAKRIAEQYQSGMRRLVVVVSAMGKETNRLVSLTESLNQPI  2699
             +IVQK+GGTSV S EKI   A R   + Q G   +VVVVSAMGK T+ LV+L   +
Sbjct  3     LIVQKFGGTSVGSAEKIQHAANRAIAEKQKG-HDVVVVVSAMGKSTDALVNLAAEITSEP  61

Query  2698  DSASYDLVVSAGEQVSAGLMAAALKQECIPAQPFLAHQLAICTDRLHGAAQINSIDINKI  2519
                   D+++S GEQV+  L+A AL+++   A  +   Q  + T+ +HG A+I  IDI  I
Sbjct  62    SKREMDMLLSTGEQVTISLLAMALQEKGYDAVSYTGWQAGVRTEAVHGNARITDIDITAI  121

Query  2518  HACWQQGKIPVVAGFQGVNDLGELTTLGRGGSDttaatlaaflnadLCEINTDVEGVYSA  2339
              A   +G+I VVAGFQGV + G +TTLGRGGSDTTA  LAA L AD C+I TDV GV++
Sbjct  122   QAQLAEGRITVVAGFQGVTEDGGITTLGRGGSDTTAVALAAALGADKCDIYTDVPGVFTT  181

Query  2338  DPNRVKGAQLLKVLDYEVAREMAVLGSRVLHPRCVEISAQYNIPIVVRDTFSKDHRDYTL  2159
             DP  V+ A+ L  + Y+   E+A LG+ VLHPR VE +  Y +P+ VR   S +H   TL
Sbjct  182   DPRYVRSARKLAGISYDEMLELANLGAGVLHPRAVEFAKNYQVPLEVRS--STEHEAGTL  239

Query  2158  VHSGVESRCKNNAQVTSLSVDKKVARVLLTAITS--KQMAEVFDCLAQAAVNVDVIVHER  1985
             +    ES  + N  V  ++ + ++ RV +T +TS    ++ +F  LA+  +NVD+I+  +
Sbjct  240   IEE--ESSMEQNLIVRGIAFEDQITRVSVTGLTSGLTTLSTIFTALAKRNINVDIIIQTQ  297

Query  1984  VEEEDKAQVSFSVSTVDQLKTLAILEK  1904
               ++ +A +SFSV T D  +T+A+LE+
Sbjct  298   A-DDGEADISFSVKTEDAAQTVAVLEE  323
```

<u>In this case, we have a bad annotation. The first protein is an Aspartokinase I and the second is an Aspartokinase II. I don't know what the difference is, but I don't think these should go together.</u>

Concluding thoughts

This is a useful exercise. I have shown that this genome has two believable gaps of 14 bp and 122 bp. However, I also showed that in two cases the annotations were not of sufficient quality to make an assertion.

Of course, these gaps are those that I could identify easily: they must have sufficient protein sequences flanking the gaps that can be recognized by kmers.

The svr scripts provide a starting point, and then those contigs identified as candidates must be compared to the nr database (or similar).

Contigs should only be joined when the following criteria are met:

1. The same protein is matched on both contigs
2. The ends of the contigs do not overlap (that is, there is a clear gap in the protein, and the combined extra DNA sequence between the end of the alignments on the DNA sequences and does not exceed the number of amino acids in the gap).
3. Care needs to be taken about strand issues.

Conjectures

Formulating Conjectures: Using the Browser and Atomic Regulons

By Ross Overbeek, March 26, 2011

We think of an *atomic regulon* as a set of genes that are tightly regulated as a unit – that is, they all are expressed at the same points in time. This is obviously a gross simplification, and maybe even an over simplification. We think that it is a useful concept, and we employ it regularly in our attempts to make sense of the genomic data and its integration with expression data.

If you are looking at a gene in the browser, you may see a little link indicating that the gene is in something we call an atomic regulon. Look at fig|300852.3.peg.1726 as an example. The line

atomic regulon membership Atomic regulon 6 **of size 13 in 300852.3**

should appear, and if you click on the link, it should take you here. This is a display that has two basic tables. The first shows Pearson Correlation Coefficients between genes in the atomic regulon (based on expression data that has been loaded into the SEED). The second table describes the genes and their current function assignments. At the time of this writing, there appear to be 10-11 genes for which the functional role is at least partially understood and 2-3 for which it is not yet clear. We suggest looking at The SoxYZ Complex Carries Sulfur Cycle Intermediates on a Peptide Swinging Arm or Structural Basis for the Oxidation of Protein-bound Sulfur by the Sulfur Cycle Molybdohemo-Enzyme Sulfane to get some insight into what is known about the genes in this cluster. It is a critical cluster including a set of genes that catalyze a key reaction in the global sulfur cycle. A domain expert could almost certainly improve the annotations.

Now, let me point you at another interesting atomic regulon. Go to the PEG page for fig|300852.3.peg.2046. Note that this is a hypothetical protein. If you go to the atomic regulon page containing it the situation becomes much clearer. The hypothetical protein probably relates to cobalamine (co-enzyme B12) synthesis.

Or again, look at the PEG page for fig|300852.3.peg.1074. The encoded protein is annotated as *NADH-FMN oxidoreductase.* When you look at the atomic regulon, things begin to get clearer: this gene (and a second gene annotated as a hypothetical protein) both participate in an aromatic amino acid degradation pathway. Again, a domain expert could almost certainly say more.

The ability to peruse these atomic regulons looking for such interesting cases is provided from the **Navigate** option on the toolbar. Pick **atomic regulons,** and then a genome with expression data and simply think about what is being displayed.

One of my earlier efforts to bring this wealth of data to the attention of some of my colleagues is displayed in http://www.theseed.org/TheBook/HTML/atomic_regulons.html

That document was produced in October, 2010, and the exact atomic regulons have changed (due to more available data). It also reflects a tool that was never released within the SEED Project and existed only as a prototype. However, my mindset (exuberance at this wealth of data) still exists, and you might find sections interesting.

Formulating Conjectures: Using the Browser and Atomic Regulons - Part 2

by Ross Overbeek, March 26, 2011

This should be thought of as a continuation of my previous short note giving examples of genes in which the notion of *atomic regulon*, along with expression data, supported the formulation of problems/hypotheses relating to the functions of gene products.

1. fig|300852.3.peg.2216: an Example Relating to CRISPRs

Try looking at peg.2216 in the SEED browser. If you expand the "compare regions" to include a few more genomes (and, you might make it a bit wider too), you see a cluster of 5 genes that occurs in several distinct genomes (note that we have duplicate versions of several of these genomes). If you look at the atomic regulon containing the gene, it looks a little bleak. There is only a regulatory protein, what appears to be a CRISPR-related protein and 3 hypotheticals. What do you make of that? First, note that some of the genes in *Cyanothece sp. PCC 8802* are clear remnants of a CRISPR event.

First, I recommend using **psi-blast** (see the link above) to verify that peg.2218 really is probably CRISPR-related. Once you have done that, work through the remaining genes using psi-blast. What you will find is that peg. 2217 is also CRISPR-associated. After a few iterations, you will also see a very distant relationship between peg.2215 and a protein annotated as CRISPR-associated.

I think that it is likely that one could put together a coherent picture of the genes in this atomic regulon, and it would center on CRISPRs. These types of examples – those in which the cluster of genes is very local to just a few genomes – are less interesting to most of our annotators. However, it is clear that at least part of the story can be revealed from just the 3-4 different genomes that are relevant.

2. fig|224911.1.peg.1749 in Bradyrhizobium japonicum USDA 110

Consider fig|224911.1.peg.1749. Now go to the atomic regulon containing it. It seems completely clear that this atomic regulon relates to nitrogen fixation in *Bradyrhizobium* and that at least two hypothetical proteins can be directly related to that process.

3. fig|224911.1.peg.1443 in Bradyrhizobium japonicum USDA 110

Consider fig|224911.1.peg.1443, and go to the atomic regulon containing it.. It seems to me that the evidence supports the hypothesis that this gene relates to a secretion system that is spread over two loci. When I showed it to one of our annotators, she commented

> This family is imbedded in// associated with // this adhesion cluster only in 2 genomes out of many that contain it – only in Bradyrhizobium – a plant symbiont (this can be seen by repositioning yourself on any other gene of the cluster and by expanding it to 35 genomes or so). Hence, it is probably not the part of core machinery. Interestingly, annotation of this family in Pfam associates it with plant cell surface was – exactly what they need to adhere to …
>
> "PF04116: This superfamily includes fatty acid and carotene hydroxylases and sterol desaturases. Beta-carotene hydroxylase is involved in zeaxanthin synthesis by hydroxylating beta-carotene, but the enzyme may be involved in other pathways [1]. This family includes C-5 sterol desaturase and C-4 sterol methyl oxidase. Members of this family are involved in cholesterol biosynthesis and biosynthesis a plant cuticular wax..."

4. fig|211586.9.peg.2693 in Shewanella oneidensis MR-1

fig|211586.9.peg.2693 encodes what seems to be a completely uninterpretable protein, at least until you look at the associated atomic regulon. Once you see that the clues point directly at a prophage. In fact, it is very common to run into a sequence of hypotheticals that appear to be quite unlike most proteins in the existing databanks, and later find via detailed analysis that you have a prophage. The speed with which phages evolve, and there abundance in the environment, lead to many, many hypotheticals that are difficult to pin down.

5. fig|211586.9.peg.2892 in Shewanella oneidensis MR-1

Look at fig|211586.9.peg.2892, and then look at the associated atomic regulon. It seems clear that peg.2891 and peg.2892 both relate to chemotaxis and motility. When I showed this to an annotator, the response was

> This family ONLY occurs in Shewanella – too narrow… Not even represented in Pfams. Wouldn't it be better to select widely distributed hypotheticals for examples? They are of more interest and also - have many more potential clues associated with them - for further digging into possible function

She is right in the sense that machinery that is very local is harder to pin down. Yet, it does seem to me that we could reasonably annotate the gene now as "*related to flagellar motility*", and it might be worth pondering how one might substantiate that claim.

Note that you have a similar situation with fig|208964.1.peg.4298 in which you can conjecture a role in pilus assembly, even though the gene is fairly local (in this case to the *Pseudomonads).*

6. fig|211586.9.peg.4166 in Shewanella oneidensis MR-1

Look at fig|211586.9.peg.4166. If you just do psi-blast, you find that the protein is very poorly characterized:

> **DUF1234**: Alpha/Beta hydrolase family of unknown function (DUF1234). The Crystal Structure Of The Yden Gene Product from B. Subtilis has been
>
> solved. The structure shows an alpha-beta hydrolase fold suggesting an
>
> enzymatic function for these proteins.

That seems like fairly minimal information. However, when you look at the atomic regulon, the picture starts to fill in. We still think that the clustering on the chromosome is probably the richest source of insight, and in this case, if you expand the compare regions to many genomes, it is worth looking at the rearrangement shown in *Acinetobacter,* and a second (less compelling) rearrangement in one of the *Burkholderia* genomes. The conjecture would be that the protein plays a role in inorganic sulfur assimilation.

7. fig|100226.1.peg.4182 in Streptomyces coelicolor A3(2)

The protein encoded by this gene is annotated as "Putative uncharacterized protein" by UniProt and as "2SCD46.40c, unknown, len: 82aa" by the SEED (really an awful estimate, but it does convey a certain lack of clues). Anyway, looking at the psi-blast output suggests that it is a "*gualylate cyclase protein*". Looking at the atomic regulon suggests a role in phosphate metabolism.

I do not claim that we can pin this down easily, but I do believe that the suggestions made (in this case) by psi-blast and atomic regulons are useful. The atomic regulons were computed from a fairly small set of experiments, so you cannot take them too seriously.

Differential Expression Analysis tool

We provide a web page for performing differential expression analysis between two replicate sets from a gene expression experiment. This analysis is similar to a fold-change analysis, in that it ranks the genes from most upregulated to most downregulated. It also performs gene set analysis on sets of genes from **subsystems** and **atomic regulons**. The web page can be accessed here:

http://bioseed.mcs.anl.gov/~dejongh/FIG/seedviewer.cgi?page=DifferentialExpression

Note that you must log in using your RAST account to use this web page.

Select a genome

The first page asks you to select a genome. If you click in the text box, you will see a drop down list of all the genomes for which we have loaded gene expression data into the SEED database. The full list of genomes and details about the experiments loaded for them are available here:

http://www.theseed.org/TheBook/GXP.htm

You can also type in the text box to filter the genomes according to the text that you enter.

Click “Load Expression Samples for Genome” to proceed.

Select expression samples

For each Replicate Set, you must select one or more samples from the drop down lists available from each text box. Again you can type in the text box to filter the samples. The gene expression values for each gene across all samples in each Replicate Set will be averaged to obtain an expression value for the gene. Every sample that starts with the prefix “GSM” is from NCBI’s GEO database, which can be searched at http://www.ncbi.nlm.nih.gov/geo/

Click “Run Analysis” to proceed.

Differential Expression Results

The results table has three tabs:

The "Subsystems" tab (green) shows the results of the gene set analysis for each Subsystem. The "Size" column shows how many pegs/rnas in the query genome are associated with each subsystem in the SEED database. The "Mean" column shows the mean differential expression value for all features in the subsystem. The two "p-value" columns show a probability estimate of how likely it is that a set of genes of the given size will have the given differential expression value. Look for the subsystems with the highest means and lowest over expressed p-values, as well as the lowest means and the lowest under expressed p-values.

The "Atomic Regulons" tab (green) shows a similar gene set analysis for each atomic regulon.

The "Genome Features" tab shows ranked differential expression results for each protein-encoding gene ("peg") and RNA in the genome. Specific peg and rna features can be searched by typing in the text box below the heading "Feature". Likewise, the "Function", "Subsystems", and "Atomic Regulon" columns are all searchable. All columns are sortable, by clicking on the up/down arrows. Since p-values could not be computed for individual features, the column "Rank out of xxxx" can be used instead to evaluate the ranking of differential expression result for each feature in the context of the entire genome

Excersise:

For a trial run of this tool you are welcome to either follow along with example on the screen or chose one of the test cases suggested here.

1. Open the list of available experiments, locate *Pseudomonas aeruginosa* PA01 and follow the provided link to GEO Platform GPL84
2. Scroll to about mid-page and open/maximize the description of the Series (58)
3. Again scroll down (or use "Find on page" functionality of your Browser) to locate series/experiment "GSE22665"
4. Click on it to open the detailed description of the experiment "Expression data from P. aeruginosa colonizing the Murine GI Tract" (PMID: 21170272). A shortcut to this GEO page: http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE22665
5. Scroll to about mid-page again and open the list of Samples (6)

* * * * * *

6. Open Differential Expression tool a separate window of your Browser.

7. From “Select a Genome” page choose *Pseudomonas aeruginosa* PA01 and press “Load Expression samples for genome” button

8. Start typing or copy/paste GEO ID for one of the Control samples (e.g. GSM562112) into “Expression Sample 1” field. Use “Add Another Sample” button to add the 2 remaining control replicate samples. They all will be averaged into “Replicate Set 1” (note, that Rep1 set is treated by this tool as a control)

9. Likewise, enter 3 replicate samples (GSM562109, etc) into “Replicate Set 2” fields

10. Hit “Run Analysis”

11. Explore the Results Table:

    a. Sort “subsystems” tab (green) by Mean differential expression value. What subsystems are preferentially expressed during gut colonization? Which are repressed? There are two ways to analyze Subsystems of interest: (i) to follow the corresponding link from the “Subsystem” tab – it will take you to the underlying SEED database, leaving the Differential Expression viewer; (ii) alternatively you can copy SS name and paste it into “Subsystems” text box under “Genome Features” tab. This way expression of each PEG or RNA associated with this SS can be explored

    b. Sort “Genome Features” tab (green) by Rank – explore

    c. Which Atomic Regulons are represented among the highest-ranking PEGs? Find them under “Atomic Regulons” tab, and follow the links

    d. Try typing “rna” in the text box below the heading “Feature”

12. The beauty of the normalization procedure that is utilized to build our microarray data compendium in that it allows comparison between any samples within the data collection, regardless of which experiment they were originally generated. For example, gene expression in sterile water can be compared to *Pseudomonas aeruginosa* planktonic growth in rich medium (sample GSM260371), and not to colonizing cells in this experiment

Some Notes on How to Look for Co-expressed Genes Using the Expression Data

There are two main SVR tools that you can use to get information relating to which other genes might be co-expressed with a given gene:

```
svr_corr_by_exp < PEGs
```

and

```
svr_coregulated_by_correspondence.pl < PEGs
```

The first tool is used only for PEGs that occur in organisms for which expression data exists in the SEED (you can find out which organisms have such data by going to the PubSEED (http://pubseed.theseed.org/seedviewer.cgi) going to the "Navigate" dropdown on the header bar, and clicking on "Atomic Regulons". This is admittedly a klutzy way to do this, but it works (since we have computed "atomic regulons" for precisely the set of organisms for which we have expression data).

Suppose that the file "in" contained

```
-------
fig|83333.1.peg.896
fig|190650.1.peg.3561
fig|316385.5.peg.980
-------
```

Then,

```
svr_corr_by_exp < in
```

would produce

```

------
fig|83333.1.peg.896     0.760   fig|83333.1.peg.23
fig|83333.1.peg.896     0.768   fig|83333.1.peg.3250
fig|83333.1.peg.896     0.762   fig|83333.1.peg.3013
fig|83333.1.peg.896     0.848   fig|83333.1.peg.4110
fig|83333.1.peg.896     0.789   fig|83333.1.peg.3110
fig|83333.1.peg.896     0.826   fig|83333.1.peg.3232
fig|83333.1.peg.896     0.850   fig|83333.1.peg.4112
fig|83333.1.peg.896     0.832   fig|83333.1.peg.3275
fig|83333.1.peg.896     0.759   fig|83333.1.peg.3255
fig|83333.1.peg.896     0.819   fig|83333.1.peg.3241
fig|83333.1.peg.896     0.783   fig|83333.1.peg.3240
```

```
fig|83333.1.peg.896     0.820   fig|83333.1.peg.3237
fig|83333.1.peg.896     0.800   fig|83333.1.peg.3231
fig|83333.1.peg.896     0.838   fig|83333.1.peg.3248
fig|83333.1.peg.896     0.887   fig|83333.1.peg.169
fig|83333.1.peg.896     0.831   fig|83333.1.peg.2576
fig|83333.1.peg.896     0.822   fig|83333.1.peg.3245
fig|83333.1.peg.896     0.697   fig|83333.1.peg.3342
fig|83333.1.peg.896     0.859   fig|83333.1.peg.3230
fig|83333.1.peg.896     0.828   fig|83333.1.peg.3174
fig|83333.1.peg.896     0.816   fig|83333.1.peg.3276
fig|83333.1.peg.896     0.723   fig|83333.1.peg.895
fig|83333.1.peg.896     0.745   fig|83333.1.peg.125
-----
```

You can use the

```
      -m MinPCC
```

parameter to designate a minimum value for the Pearson Correlation Coefficient. Thus,

```
      svr_corr_by_exp -m 0.8 < in | svr_function_of
```

would produce

```
-----
fig|83333.1.peg.896     0.848   fig|83333.1.peg.4110    SSU ribosomal protein S6p
fig|83333.1.peg.896     0.826   fig|83333.1.peg.3232    SSU ribosomal protein S13p
(S18e)
fig|83333.1.peg.896     0.850   fig|83333.1.peg.4112    SSU ribosomal protein S18p
fig|83333.1.peg.896     0.832   fig|83333.1.peg.3275    SSU ribosomal protein S7p
(S5e)
fig|83333.1.peg.896     0.819   fig|83333.1.peg.3241    SSU ribosomal protein S14p
(S29e)
fig|83333.1.peg.896     0.820   fig|83333.1.peg.3237    SSU ribosomal protein S5p
(S2e)
fig|83333.1.peg.896     0.800   fig|83333.1.peg.3231    SSU ribosomal protein S11p
(S14e)
fig|83333.1.peg.896     0.838   fig|83333.1.peg.3248    SSU ribosomal protein S3p
(S3e)
fig|83333.1.peg.896     0.887   fig|83333.1.peg.169     SSU ribosomal protein S2p
(SAe)
fig|83333.1.peg.896     0.831   fig|83333.1.peg.2576    SSU ribosomal protein S16p
fig|83333.1.peg.896     0.822   fig|83333.1.peg.3245    SSU ribosomal protein S17p
(S11e)
fig|83333.1.peg.896     0.859   fig|83333.1.peg.3230    SSU ribosomal protein S4p
(S9e)
fig|83333.1.peg.896     0.828   fig|83333.1.peg.3174    SSU ribosomal protein S9p
(S16e)
fig|83333.1.peg.896     0.816   fig|83333.1.peg.3276    SSU ribosomal protein S12p
(S23e)
-----
```

You may want to estimate co-regulated genes in a case where you have no expression data for a gene. You can gain some insight by

1. mapping your given gene to corresponding genes in organisms with expression data,

2. gathering the genes that the corresponding gene maps to, and

3. mapping that gene back to a corresponding gene in the initial genome.

We call this "indirect estimate of co-expression". You can gather such indirect evidence from all genomes with expression data, and it often is quite useful.

You might try

```
    svr_coregulated_by_correspondence.pl < in
```

It should produce lines like

```
-----
fig|316385.5.peg.980    fig|312309.3.peg.1759,0.763,fig|312309.3.peg.1757
fig|316385.5.peg.1382
------
```

You may think of this as saying "fig|316385.5.peg.980 may have correlated expression with fig|316385.5.peg.1382 due to evidence given in the middle column.

This middle column says that

1. fig|312309.3.peg.1759 seems to correspond to fig|316385.5.peg.980,

2. fig|312309.3.peg.1757 seems to correspond to fig|316385.5.peg.1382, and

3. fig|312309.3.peg.1759 has expression values that show a Pearson coefficient of correlation of 0.763 with fig|312309.3.peg.1757,

Most of the lines would contain multiple components of evidence, each being a comma-separated 3-tuple. You may find things a bit clearer if you asked for the evidence to be shown as separate fields, along with the function of the PEGs. This can be done using the -f option. Thus,

```
   svr_coregulated_by_correspondence -f < in
```

produces lines like (Each line of output is displayed as four, below)

```
-----
fig|83333.1.peg.896     SSU ribosomal protein S1p
fig|100226.1.peg.1964   0.693   tRNA(Ile)-lysidine
synthetase   fig|100226.1.peg.3366   tRNA(Ile)-lysidine
synthetase   fig|83333.1.peg.188

fig|83333.1.peg.896     SSU ribosomal protein S1p
fig|226186.1.peg.4343   0.651   tRNA(Ile)-lysidine
```

```
synthetase   fig|226186.1.peg.1593   tRNA(Ile)-lysidine
synthetase   fig|83333.1.peg.188

fig|83333.1.peg.896     SSU ribosomal protein S1p
fig|312309.3.peg.1759   0.701   tRNA(Ile)-lysidine
synthetase   fig|312309.3.peg.1944   tRNA(Ile)-lysidine
synthetase   fig|83333.1.peg.188

fig|83333.1.peg.896     SSU ribosomal protein S1p
fig|312309.3.peg.1759   0.861   tRNA(Ile)-lysidine
synthetase   fig|312309.3.peg.1945   tRNA(Ile)-lysidine
synthetase   fig|83333.1.peg.188
-----
```

You can use the

```
    -m MinPCC
```

parameter to force the correlation values to all be greater than or equal to MinPCC. Finally, there is an extra consideration worth mentioning: many genes within a genome have large correlation values with a large number of other genes simply because they are all always ON or always OFF for the set of experiments we have included. It is usually not useful to include such correspondences, and we suggest using

```
   -n 50
```

which says "only consider corresponding PEGs useful is they have strong correspondences with 50 or fewer other genes". In fact, we have set the default to that value, so you would use

```
   -n 10000
```

to get everything (which can be quite sizable).

A Case Study in Use of the "Server Scripts"

The existing SEED implementation supports programmatic APIs to a set of servers. In the following example, we show a pipeline of little utility scripts, each accessing a server over the network to extract data. The cumulative function achieved by composing these scripts is worth considering in some detail. The pipeline of commands that we use in this case study is as follows:

```
svr_gap_filled_reactions_and_roles -m 214092.1 | cut -f1,2
| sort -u |
svr_find_clusters_relevant _to_reaction -g 1 –c 2 |
svr_find_hypos_for_cluster |
svr_missing_roles -g 1 -r 2
```

The first line access the Model SEED server to acquire the list of reactions that were added to the initial model for genome 214092.1 in the process of “gap filling”. That is, these are the reactions and functional roles that were believed to be needed, but for which no specific genes had been connected to the functional roles. Since we wished just the reactions (ignoring the functional roles), we kept only the first two fields in the output and removed duplicate lines.

The second line takes the input lines, each corresponding to a gap-filled reaction, and looks for clusters of genes on the chromosome. It determines the functional roles needed to implement the gap-filled reaction and reactions that neighbor it in the metabolic network. Then it looks for clusters of genes (by annotation) that implement two or more of those functional roles and occur close to one another on the chromosome. Each output line describes a cluster of genes in the genome of 214092.1 that represent distinct, but perhaps related, functional roles based on the programs awareness of the abstract reaction network.

The third line of the command invokes a tool that looks for genes with unknown function that are associated with the cluster on each input line. Each line of output will contain a gap-filled reaction, a set of genes that apparently cluster, and the ID of a gene of unknown function.

Finally, the fourth line of the command looks for functional roles that are needed for the gap-filled reaction, but for which no genes have yet been connected. The set of these roles are added to each input line to produce a final output line.

Thus, each output line represents a single gene of unknown function that occurs close to a cluster of genes that perform functions relating to reactions that are “close in the metabolic network”. Further, the output line contains a set of functional roles that need to be connected to genes to support the gap-filled reaction. This becomes an input packet given to a human to formulate and analyze which of the possible conjectures is most likely.

Finally, it is worth noting that we could just alter the pipeline slightly, using

```
svr_all_models |
svr_gap_filled_reactions_and_roles -c 1 |
cut -f1,3 | sort -u |
svr_find_clusters_relevant_to_reaction -g 1 -c 2 |
svr_find_hypos_for_cluster |
svr_missing_roles -g 1 -r 2
```

and the pipeline would then produce hypotheticals occurring in clusters that form candidates for genes implementing gap-filling reactions for all reactions added to the entire collection of hundreds of metabolic models. That is, it would in effect perform the simple exploratory steps for every genome for which we have built a metabolic model.

Searching for UDP-2,3-diacylglucosamine hydrolase (EC 3.6.1.-) in *Agrobacterium tumefaciens str. C58*

Gap-filling in Agrobacter tumefacians str. C58 (176299.3) led to the prediction that the reaction **rxn03130**, requiring the functional role

UDP-2,3-diacylglucosamine hydrolase (EC 3.6.1.-)

was probably active. The use of the scripts

```
svr_find_clusters_relevant_to_reaction
svr_hypos__for_cluster
svr_missing_roles
```

led to the following observations:

There is a cluster of PEGs from peg.1723 through peg.1727 that contains a number of genes assigned functions that close in the reaction network to the sought hydrolase:

peg.1723: UDP-3-O-[3-hydroxymyristoyl] glucosamine N-acyltransferase (EC 2.3.1.-)

peg.1724: (3R)-hydroxymyristoyl-[acyl carrier protein] dehydratase (EC 4.2.1.-)

peg.1725: Acyl-[acyl-carrier-protein]--UDP-N-acetylglucosamine O-acyltransferase (EC 2.3.1.129)

peg.1727: Lipid-A-disaccharide synthase (EC 2.4.1.182)

peg.1726 is embedded in the cluster. It currently is annotated as Protein of unknown function DUF1009 clustered with KDO2-Lipid A biosynthesis genes

The clustering of peg.1726 with genes having these functions suggests it as a candidate for our "missing gene".

The functional coupling data for these genes showed

Gene	OTUs	Coupled to	Function of coupled-to Gene
fig|176299.3.peg.1724	179	fig|176299.3.peg.1725	Acyl-[acyl-carrier-protein]--UDP-N-acetylglucosamine O-acyltransferase
fig|176299.3.peg.1723	179	fig|176299.3.peg.1724	(3R)-hydroxymyristoyl-[acyl carrier protein] dehydratase (EC 4.2.1.-)
fig|176299.3.peg.1725	145	fig|176299.3.peg.1722	Outer membrane protein assembly factor YaeT precursor
fig|176299.3.peg.1723	145	fig|176299.3.peg.1722	Outer membrane protein assembly factor YaeT precursor
fig|176299.3.peg.1724	138	fig|176299.3.peg.1722	Outer membrane protein assembly factor YaeT precursor
fig|176299.3.peg.1727	130	fig|176299.3.peg.1723	UDP-3-O-[3-hydroxymyristoyl] glucosamine N-acyltransferase
fig|176299.3.peg.1725	130	fig|176299.3.peg.1727	Lipid-A-disaccharide synthase (EC 2.4.1.182)
fig|176299.3.peg.1723	130	fig|176299.3.peg.1727	Lipid-A-disaccharide synthase (EC 2.4.1.182)
fig|176299.3.peg.1723	130	fig|176299.3.peg.1725	Acyl-[acyl-carrier-protein]--UDP-N-acetylglucosamine O-acyltransferase
fig|176299.3.peg.1724	121	fig|176299.3.peg.1727	Lipid-A-disaccharide synthase (EC 2.4.1.182)
fig|176299.3.peg.1724	106	fig|176299.3.peg.1721	Membrane-associated zinc metalloprotease
fig|176299.3.peg.1727	103	fig|176299.3.peg.1722	Outer membrane protein assembly factor YaeT precursor
fig|176299.3.peg.1725	103	fig|176299.3.peg.1721	Membrane-associated zinc metalloprotease
fig|176299.3.peg.1723	103	fig|176299.3.peg.1721	Membrane-associated zinc metalloprotease
fig|176299.3.peg.1725	58	fig|176299.3.peg.1726	unknown function - clustered with KDO2-Lipid A biosynthesis genes
fig|176299.3.peg.1726	44	fig|176299.3.peg.1727	Lipid-A-disaccharide synthase (EC 2.4.1.182)
fig|176299.3.peg.1723	28	fig|176299.3.peg.1720	Phosphatidate cytidylyltransferase (EC 2.7.7.41)
fig|176299.3.peg.1723	20	fig|176299.3.peg.1726	unknown function - clustered with KDO2-Lipid A biosynthesis genes
fig|176299.3.peg.1726	14	fig|176299.3.peg.1721	Membrane-associated zinc metalloprotease
fig|176299.3.peg.1727	12	fig|176299.3.peg.1733	Citrate synthase (si) (EC 2.3.3.1)
fig|176299.3.peg.1723	10	fig|176299.3.peg.1719	Undecaprenyl pyrophosphate synthetase (EC 2.5.1.31)
fig|176299.3.peg.1724	9	fig|176299.3.peg.1720	Phosphatidate cytidylyltransferase (EC 2.7.7.41)

If you look at blast scores for peg.1726, most are “hypothetical protein”, but one is given as

UDP-2,3-diacylglucosamine pyrophosphatase [Caulobacter crescentus CB15]

>gi|221234924|ref|YP_002517360.1| UDP-2,3-diacylglucosamine pyrophosphatase [Caulobacter crescentus NA1000]

The hit is 94% identity and a psc of 2e-107

So, this leads to the conjecture that fig|176299.3.peg.1726 encodes a protein with function UDP-2,3-diacylglucosamine hydrolase (EC 3.6.1.-).

“Lipopolysaccharide core biosynthesis protein RfaZ” in *Pseudomonas aeruginosa PAO1*

Ross Overbeek, March, 2011

I began by using a tool to explore gap-filled reactions in *Pseudomonas aeruginosa PA01*, which in the SEED has an ID of 208964.1. My tool told me that reaction *rxn08954* was gap-filled, and I should begin looking for the role

> Lipopolysaccharide core biosynthesis protein RfaZ

as a needed, but unidentified gene. The tool suggested that I look in two clusters. Specifically, I was pointed at

1. fig|208964.1.peg.2980 in the cluster containing
    a. fig|208964.1.peg.2979: 3-deoxy-manno-octulosonate cytidylyltransferase (EC 2.7.7.38)
    b. fig|208964.1.peg.2981: Tetraacyldisaccharide 4'-kinase (EC 2.7.1.130)
2. fig|208964.1.peg.5004 in the cluster containing
    a. fig|208964.1.peg.5007 :UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
    b. fig|208964.1.peg.5008: Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
    c. fig|208964.1.peg.5009: ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)

Let's begin by considering the first cluster. The current annotation (in the SEED) for

fig|208964.1.peg.2980 is "UPF0434 protein YcaR".

Note that it is not RfaZ. This can be deduced (or suggested, perhaps) by the fact that E.col has both genes (ycaR and rfaZ). There is little known about the protein YcaR. However, if you look at the blast hits, it is often annotated as

tetraacyldisaccharide 4'-kinase

Off hand, it looks much too short, but do note that the gene adjacent has the same annotation, and if you use the "compare regions" tool, you will see that fig|335283.5.peg.2471 in *Nitrosomonas eutropha C91* is annotated as Tetraacyldisaccharide 4'-kinase (EC 2.7.1.130) / UPF0434 protein YcaR

That is, someone on our team believed that it is a potential gene fusion. fig|208964.1.peg.2980 co-occurs in 69 OTUs with fig|208964.1.peg.2981: Tetraacyldisaccharide 4'-kinase (EC 2.7.1.130)

Here are the functional coupling values produced by creating a file called *pegs* containing

fig|208964.1.peg.2979

fig|208964.1.peg.2980

fig|208964.1.peg.2981

And running

```
svr_functionally_coupled < pegs | svr_function_of
```

fig|208964.1.peg.2979 11 fig|208964.1.peg.2976 Ribonuclease E (EC 3.1.26.12)

fig|208964.1.peg.2979 11 fig|208964.1.peg.2977 UDP-N-acetylenolpyruvoylglucosamine reductase (EC 1.1.1.158)

fig|208964.1.peg.2979 18 fig|208964.1.peg.2978 Low molecular weight protein tyrosine phosphatase (EC 3.1.3.48)

fig|208964.1.peg.2979 56 fig|208964.1.peg.2980 UPF0434 protein YcaR

fig|208964.1.peg.2979 74 fig|208964.1.peg.2981 Tetraacyldisaccharide 4'-kinase (EC 2.7.1.130)

fig|208964.1.peg.2979 32 fig|208964.1.peg.2982 Biopolymer transport protein ExbD/TolR

fig|208964.1.peg.2979 34 fig|208964.1.peg.2983 MotA/TolQ/ExbB proton channel family protein

fig|208964.1.peg.2979 39 fig|208964.1.peg.2984 DNA internalization-related competence protein ComEC/Rec2

fig|208964.1.peg.2979 7 fig|208964.1.peg.2985 COGs COG3216

fig|208964.1.peg.2980 7 fig|208964.1.peg.2977 UDP-N-acetylenolpyruvoylglucosamine reductase (EC 1.1.1.158)

fig|208964.1.peg.2980	56	fig|208964.1.peg.2979	3-deoxy-manno-octulosonate cytidylyltransferase (EC 2.7.7.38)
fig|208964.1.peg.2980	69	fig|208964.1.peg.2981	Tetraacyldisaccharide 4'-kinase (EC 2.7.1.130)
fig|208964.1.peg.2980	22	fig|208964.1.peg.2982	Biopolymer transport protein ExbD/TolR
fig|208964.1.peg.2980	22	fig|208964.1.peg.2983	MotA/TolQ/ExbB proton channel family protein
fig|208964.1.peg.2980	36	fig|208964.1.peg.2984	DNA internalization-related competence protein ComEC/Rec2
fig|208964.1.peg.2981	10	fig|208964.1.peg.2976	Ribonuclease E (EC 3.1.26.12)
fig|208964.1.peg.2981	12	fig|208964.1.peg.2977	UDP-N-acetylenolpyruvoylglucosamine reductase (EC 1.1.1.158)
fig|208964.1.peg.2981	16	fig|208964.1.peg.2978	Low molecular weight protein tyrosine phosphatase (EC 3.1.3.48)
fig|208964.1.peg.2981	74	fig|208964.1.peg.2979	3-deoxy-manno-octulosonate cytidylyltransferase (EC 2.7.7.38)
fig|208964.1.peg.2981	69	fig|208964.1.peg.2980	UPF0434 protein YcaR
fig|208964.1.peg.2981	36	fig|208964.1.peg.2982	Biopolymer transport protein ExbD/TolR
fig|208964.1.peg.2981	34	fig|208964.1.peg.2983	MotA/TolQ/ExbB proton channel family protein
fig|208964.1.peg.2981	49	fig|208964.1.peg.2984	DNA internalization-related competence protein ComEC/Rec2
fig|208964.1.peg.2981	27	fig|208964.1.peg.2985	COGs COG3216
fig|208964.1.peg.2981	14	fig|208964.1.peg.2986	Lipoprotein releasing system transmembrane protein LolC

The obvious conjecture would be that fig|208964.1.peg.2980 forms a complex with fig|208964.1.peg.2981, and together they implement

> Tetraacyldisaccharide 4'-kinase (EC 2.7.1.130)

You might well do a little more digging before investing the wet-lab effort needed to confirm or reject the hypothesis, but at this stage it is a reasonable conjecture.

Now let's proceed to fig|208964.1.peg.5004 and see if that might encode *RfaZ.* Using blast to take the *RfaZ* from *E.coli* and search for it in *Pseudomonas aeruginosa PAO1* will convince you that there is no obvious instance of the gene. As a passing note, one of the few actual papers discussing *RfaZ* makes the following assertion

> "mutations in rfaS and rfaZ result in changes in the LPS core but do not affect the attachment of O antigen. We propose that these genes are involved in an alternative pathway for the synthesis of rough LPS species which are similar to lipooligosaccharides of other species and which are not substrates for O-antigen attachment."

Role of Escherichia coli K-12 rfa genes and the rfp gene of Shigella dysenteriae 1 in generation of lipopolysaccharide core heterogeneity and attachment of O antigen.

By Klena JD, et al, J. Bact, 1992 Nov;174(22):7297-307.

Now, if you use the domain analysis tools provided by NCBI, you will find that fig|208964.1.peg.5004 hist a domain described as:

Pfam06293: **Kdo**

Lipopolysaccharide kinase (Kdo/WaaP) family

These lipopolysaccharide kinases are related to protein kinases pfam00069. This family includes waaP (rfaP) gene product is required for the addition of phosphate to O-4 of the first heptose residue of the lipopolysaccharide (LPS) inner core region. It has previously been shown that WaaP is necessary for resistance to hydrophobic and polycationic antimicrobials in E. coli and that it is required for virulence in invasive strains of S. enterica

The gene almost certainly plays a role in the synthesis of a lipopolysaccharide. For a summary of what that might mean, you might look at

Subcell Biochem. 2010;53:69-99.
The diversity of the core oligosaccharide in lipopolysaccharides.

Silipo A, Molinaro A.

Abstract

Bacterial lipopolysaccharides (LPSs) are the major component of the outer membrane of Gram-negative bacteria. They have a structural role since they contribute to the cellular rigidity by increasing the strength of cell wall and mediating contacts with the external environment that can induce structural changes to allow life in different conditions. Furthermore, the low permeability of the outer membrane acts as a barrier to protect bacteria from host-derived antimicrobial compounds. Lipopolysaccharides are amphiphilic macromolecules generally comprising three defined regions distinguished by their genetics, structures and function: the lipid A, the core oligosaccharide and a polysaccharide portion, the O-chain. In some Gram-negative bacteria LPS can

terminate with the core portion to form rough type LPS (R-LPS, LOS). The core oligosaccharide is an often branched and phosphorylated heterooligosaccharide with less than fifteen sugars, more conserved in the inner region, proximal to the lipid A, and often carrying non-stoichiometric substitutions leading to variation and micro-heterogeneity. The core oligosaccharide contributes to the bacterial viability and stability of the outer membrane, can assure the serological specificity and possesses antigenic properties.

PMID: 20593263

I would also note that fig|208964.1.peg.5003 also has a much weaker hit against Pfam06293: Kdo (see above). It is currently annotated as a

Serine/threonine protein kinase

It would probably be better to call both fig|208964.1.peg.5003, and fig|208964.1.peg.5004

Lipopolysaccharide kinase (Kdo/WaaP) family

Now, let us step back and look at the functional coupling scores for the genes in this cluster. That is, build a file of PEG IDs and run

```
svr_functionally_coupled < pegs | svr_function_of
```

which should produce something like:

fig|208964.1.peg.5003	8	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5003	8	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5003	8	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5003	7	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5003	39	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5003	39	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5004	6	fig|208964.1.peg.5001	Glycosyl transferase in large core OS assembly cluster
fig|208964.1.peg.5004	8	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5004	8	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5004	13	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5004	13	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5004	13	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5005	8	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5005	8	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5005	8	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase

fig|208964.1.peg.5005	13	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5005	13	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5005	13	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5006	6	fig|208964.1.peg.5001	Glycosyl transferase in large core OS assembly cluster
fig|208964.1.peg.5006	8	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5006	8	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5006	13	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5006	13	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5006	13	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5007	7	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5007	13	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5007	13	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5007	13	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5007	29	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5007	29	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5008	16	fig|208964.1.peg.5002	Carbamoyltransferase in large core OS assembly cluster
fig|208964.1.peg.5008	39	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5008	13	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5008	13	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5008	13	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5008	29	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5008	30	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5008	8	fig|208964.1.peg.5011	glutamate-ammonia-ligase adenylyltransferase
fig|208964.1.peg.5009	39	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5009	13	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5009	13	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5009	13	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5009	29	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5009	30	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5009	14	fig|208964.1.peg.5010	Branched-chain amino acid aminotransferase (EC 2.6.1.42)
fig|208964.1.peg.5009	8	fig|208964.1.peg.5011	glutamate-ammonia-ligase adenylyltransferase

Similarly, it is worth a minute to look at the expression correlation scores using

```
svr_corr_by_exp < pegs | svr_function_of
```

which should give something like:

fig|208964.1.peg.5003	0.859	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5003	0.807	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5003	0.778	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5004	0.771	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5004	0.776	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5004	0.778	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5005	0.762	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5005	0.771	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5005	0.859	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5006	0.762	fig|208964.1.peg.5005	Heptose kinase WapQ, eukaryotic type
fig|208964.1.peg.5006	0.687	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5006	0.781	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5006	0.776	fig|208964.1.peg.5004	hypothetical protein
fig|208964.1.peg.5006	0.807	fig|208964.1.peg.5003	Serine/threonine protein kinase
fig|208964.1.peg.5007	0.723	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5007	0.714	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5007	0.781	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5008	0.723	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)
fig|208964.1.peg.5008	0.723	fig|208964.1.peg.5009	ADP-heptose--lipooligosaccharide heptosyltransferase II (EC 2.4.1.-)
fig|208964.1.peg.5008	0.687	fig|208964.1.peg.5006	Lipopolysaccharide core biosynthesis protein WaaP (EC 2.7.-.-), heptosyl-I-kinase
fig|208964.1.peg.5009	0.723	fig|208964.1.peg.5008	Lipopolysaccharide heptosyltransferase I (EC 2.4.1.-)
fig|208964.1.peg.5009	0.714	fig|208964.1.peg.5007	UDP-glucose:(heptosyl) LPS alpha1,3-glucosyltransferase WaaG (EC 2.4.1.-)

This will need to be cleaned up by an expert in lipopolysaccharide synthesis.

In any event, this little example should suggest that we might need to look at what rxn08954 is really trying to accomplish and whether or not

Lipopolysaccharide core biosynthesis protein RfaZ

Is the only role that should be thought of as implementing the reaction? In fact, the reaction is

CMP-3-deoxy-D-manno-octulosonate + l-heptosyl-kdo2-lipidA

<=>

CMP + tosyl-heptosyl-kdo2-lipidA

and the description of the enzyme is

3-deoxy-D-manno-octulosonic acid transferase III (LPS core biosynthesis)

These do not seem to match what EcoCyc identifies as the function: "*protein involved in KdoIII attachment during lipopolysaccharide core biosynthesis".* It is quite possible that the detailed variations needed to accurately characterize the alternative processes involved in LPS core biosynthesis will not be really accurately captured for some time.

So, this short report on my perusal of trying to pursue leads relating to finding "missing genes" reveals a great deal of imprecision and potential for getting things wrong. It also illustrates the wealth of data that is emerging and which will, hopefully, be used to support experts in their efforts to get it right**.**

An Exercise in Tools for Formulating Conjectures

By Ross Overbeek

March 10, 2011

Introduction

The SEED now has metabolic models for hundreds of genomes, and very soon all of the complete prokaryotic genomes (thousands of them) in the SEED will have initial metabolic models. The effort to build these has been led by Chris Henry and is described elsewhere [see High-throughput generation, optimization and analysis of genome-scale metabolic models]. One offshoot of this effort involves trying to find genes that implement the functions that were needed to construct a workable metabolic network, but for which no genes have yet been associated. That is, the process of *gap-filling* produces conjectures of the form "*A gene implementing functional role X must exist within the genome*".

Let us consider the genome 176299.3, which is *Agrobacterium tumefaciens str. C58*. If we run

```
svr_gap_filled_reactions_and_roles -m 176299.3| cut -f1,2 | sort -u |
svr_find_clusters_relevant_to_reaction -g 1 -c 2 |
svr_find_hypos_for_cluster |
svr_missing_roles -g 1 -r 2
```

we get the following file as output.

You may want to run the commands one-at-a-time saving the intermediate output in files.

To understand how we created such a command line, you need to understand the commands that make up this little pipeline. The first line,

```
svr_gap_filled_reactions_and_roles -m 176299.3| cut -f1,2 | sort -u
```

writes as output a 2-column tab-separated table. That is, each line contains 2 values separated by a tab. The first value is the genome (176299.3 in this case). The second is the ID of a reaction that was added to the initial metabolic model as a result of the gap-filling algorithm. That is, this reaction is believed to be active in the cell, but we have not yet been able to connect all of the needed functional roles to precise genes.

The second line reads in lines produced by the first and extends them a single column. This added column is a set of genes (comma-separated) that together make up a cluster of close genes that implement functional roles that are closely related to the gap-filled reaction. That is, we have computed the reaction network, and the roles implemented by the genes in the cluster are all "close to the gap-filled reaction in the metabolic network". Thus, this cluster represents what we think of as a good place to search for genes that might implement any roles that are needed but not yet found.

The third command adds two more fields to each line. The first is what we call *the functional coupling score,* and the second is a gene that is currently considered "hypothetical" (sometimes our estimate of what is currently hypothetical is a bit off – it is hard to look at a function and determine whether or not a precise function has yet been assigned to the gene). Further, that gene is within the boundaries of the cluster, or it is a gene that tends to co-occur close to one of the genes in the cluster. If the gene tends to co-occur, then the functional coupling score will give the number of times we have seen them co-occur in genomes that are somewhat distant (technically, the number of distinct *operational taxonomic units (OTUs)* in which they co-occur). Finally, the fourth line

```
svr_missing_roles -g 1 -r 2
```

takes each line and computes the set of functional roles that have not yet been bound to genes, but are needed to support the gap-filled reaction. Thus, we now have a hypothethical protein (or, more precisely, a gene encoding a protein with unknown function) and an associated set of roles that we are trying to bind to genes.

The gene **figl176299.3.peg.1726** occurs in several lines of the output. Suppose that you wanted to also get the details on the genes that **figl176299.3.peg.1726** co-occurs with. To see this, use something like

```
svr_functionally_coupled < file.containing.pegs
```

where *file.containing.pegs* is just a file containing lines with gene IDs (in this case just one line containing **figl176299.3.peg.1726** would do nicely). You would get the following lines as output:

fig|176299.3.peg.1726	15	fig|176299.3.peg.1721
fig|176299.3.peg.1726	40	fig|176299.3.peg.1722
fig|176299.3.peg.1726	71	fig|176299.3.peg.1723
fig|176299.3.peg.1726	49	fig|176299.3.peg.1724
fig|176299.3.peg.1726	71	fig|176299.3.peg.1725
fig|176299.3.peg.1726	56	fig|176299.3.peg.1727

Note that the gene is quite clearly co-occurring with the entire set of genes in the area originally suggested by the two genes in the original cluster.

This has been a minimal exposure to some fairly powerful tools. If you take the time to select a genome with which you have some familiarity, and you run the same tools we presented, you can spend hours searching for a consistent interpretation.

Finding the Neighborhood of a Reaction/Role

There are two simple commands that you can use to explore the neighborhood around a reaction. First, suppose that you have a given reaction (say, 'rxn02003'). Then

```
svr_neighboring_reactions -r rxn00979 -d 1
```

will print a table of the reactions that are immediate neighbors of the designated role (in the metabolic network).

The first column is the starting reaction, the second is the number of steps taken to reach the reaction shown in the third column. To get more than just the immediate neighbors, you can use the "-d N" parameter. Thus, "-d 2" would give you all of the functional roles that could be reached in two or less steps.

The second simple tool allows you to see the details of a reaction. Thus,

```
svr_reaction_description –r rxn02270 -n
```

would display

rxn02270 [cpd00015: Flavinadeninedinucleotideoxidized] + (cpd00760: M_2_Methyl_butyryl_CoA) =>

(cpd00982: Flavinadeninedinucleotidereduced) + (cpd02125: 2-Methylcrotanoyl-CoA)

You can use this simple tool in a pipeline, in which case the reaction ID is taken from a column of the input table, and the reaction description is appended as a new column.

Inverting the Approach

The first part of this document illustrated an approach that began with taking a gap-filled reaction and looking for hypothetical proteins that might implement the needed functional role. If you have found a hypothetical, most often it actually does not implement the role (the precision with which we pick them out is not all that great). But, what you do have is a hypothetical that is functionally bound to a cluster on the chromosome, and sometimes it makes sense to look at "What are the most likely alternatives for the function of the hypothetical?"

For example, let us start with the following output line from

svr_gap_filled_reactions_and_roles

(I have placed each field on a separate line for readability):

176299.3

rxn03130

fig|176299.3.peg.1723,fig|176299.3.peg.1727

71

fig|176299.3.peg.1726

UDP-2,3-diacylglucosamine hydrolase (EC 3.6.1.-)

Now, we might well ask "What are the most attractive alternatives for the function of peg.1726?" One simple way to get information is to just use

```
grep 'fig|176299.3.peg.1726' file.of.output.from.svr_gap_filled_reactions_and_roles
```

which produces two lines: the one above and a similar one indicating another possible functional role:

Lipid A biosynthesis (KDO) 2-(lauroyl)-lipid IVA acyltransferase (EC 2.3.1.-)

That is, the crude pipeline we implemented in this little write-up has suggested two possible functions for **fig|176299.3.peg.1726,** both of which are believed to be needed to support a working metabolic network. We believe that some modest work will lead pretty rapidly to the conclusion that the real function is probably

UDP-2,3-diacylglucosamine hydrolase (EC 3.6.1.-)

We suggest starting with psi-blast, if you wish to pursue the conjecture.

Analysis of Metagenomics using the SEED

Getting Summaries of Functional Content and OTUs for an Metagenomic Sample

(Note: in order to use the svr commands, you must have installed the myRAST app and set your environment correctly, see this post for instructions)

It is worth mentioning that two of the svr functions provide a means of getting quick summaries of content for a newly sequenced metagenomic sample.

```
svr_assign_to_dna_using_figfams < MG.sample
```

takes as input a set of DNA sequences in fasta format. It outputs a 5-column, tab-separated table containing:

1. The ID of one of the sequences
2. The number of Kmer hits against the sequence
3. The region identified as potentially supporting the function (in the form of a contig, begin, and end coordinates separated by underscores),
4. The function associated with the region (which may just be "hypothetical protein"),
5. A genome name that represents an "operational taxonomic unit" that appears to be the source of the hit.

This tab-separated table can be summarized using

```
 svr_summarize_MG_output < table > function.summary 2>
otu.summary
```

Normally, these are just pipelined using

```
svr_assign_to_dna_using_figfams < MG.sample |
svr_summarize_MG_output > function.summary 2> otu.summary
```

The pipeline will usually process roughly 6-8 megabases of data per minute.

Finally, you can use

```
 svr_metabolic_reconstruction < function.summary | cut -f
4,5 | sort -u
```

to get a quick metabolic reconstruction summarizing the active subsystems that could be determined (along with the appropriate variant code).

An Etude Relating to a Metagenomics Sample

In another short note, we suggested using

```
svr_assign_to_dna_using_figfams < MG.sample |
svr_summarize_MG_output > function.summary 2> otu.summary
```

to get summaries of function and population from a sample in the file MG.sample.

A participant in one of our tutorials took a sample and ran it through both the two commands above and did a more thorough analysis using MG-RAST. The estimates of the most highly represented phylogenetic groups differed somewhat, and the participant asked me to look into it.

I am going to use this request as a motivation for showing how one might use the server scripts effectively.

To begin with, I suggest getting a feel for how the server scripts might assign function and OTUs to data we think we understand. I decided to take the contigs of E.coli K12 and run them through the `svr_assign_to_dna_using_figfams` tool to see what came out.

This requires that I get a fasta file of the E.coli contigs. To do that, I used

```
  svr_contigs_in_genome < e.coli.id | svr_dna_seq -fasta >
ecoli.contigs
```

This constructs a file of the contigs that make up the E.coli genome.

Then I ran

```
     svr_assign_to_dna_using_figfams -by_location <
ecoli.contigs > kmer.dna.output.default
```

This produces the output of the tool on a piece of DNA that is well annotated. I asked to have the proposed hits displayed in order by location so that I could compare the proposed hits (and corresponding OTUs) against the usual annotations of the E.coli K12 genome.

The first few lines of output were as follows:

```
Contig        # hits    location (region)       proposed
function               representative of an OTU

NC_000913 14 NC_000913_190_253 Thr operon leader peptide
Escherichia coli K12
NC_000913 815 NC_000913_248_2797 Aspartokinase (EC 2.7.2.4)
/ Homoserine dehydrogenase (EC 1.1.1.3) Escherichia coli
K12
NC_000913 5 NC_000913_572_1625 hypothetical protein
Anaeromyxobacter sp. K
NC_000913 305 NC_000913_2801_3728 Homoserine kinase (EC
2.7.1.39) Escherichia coli K12
NC_000913 4 NC_000913_3213_3180 Glycerol kinase (EC
2.7.1.30) Mycobacterium smegmatis str. MC2 155
NC_000913 420 NC_000913_3734_5018 Threonine synthase (EC
4.2.3.1) Escherichia coli K12
NC_000913 136 NC_000913_5088_6022 hypothetical protein
Escherichia coli K12
NC_000913 5 NC_000913_5600_5564 Ribonucleotide reductase of
class Ia (aerobic), beta subunit (EC 1.17.4.1) Escherichia
coli K12
NC_000913 100 NC_000913_6245_2551 hypothetical protein
Escherichia coli K12
NC_000913 249 NC_000913_6459_5685 UPF0246 protein YaaA
Escherichia coli K12
NC_000913 5 NC_000913_6967_7223 hypothetical protein
Azoarcus sp. EbN1
NC_000913 3 NC_000913_7893_8614 hypothetical protein
Tsukamurella paurometabola DSM 20162
```

To get the standard annotations for E.coli, I ran

```
    svr_all_features 83333.1 peg | svr_function_of |
svr_gene_data -c 1 location > ecoli.pegs
```

The first few lines of output were

```
fig|83333.1.peg.1 Thr operon leader peptide
83333.1:NC_000913_190+66
fig|83333.1.peg.2 Aspartokinase (EC 2.7.2.4) / Homoserine
dehydrogenase (EC 1.1.1.3) 83333.1:NC_000913_337+2463
fig|83333.1.peg.3 Homoserine kinase (EC 2.7.1.39)
83333.1:NC_000913_2801+933
fig|83333.1.peg.4 Threonine synthase (EC 4.2.3.1)
83333.1:NC_000913_3734+1287
fig|83333.1.peg.5 hypothetical protein
83333.1:NC_000913_5234+297
fig|83333.1.peg.6 UPF0246 protein YaaA
83333.1:NC_000913_6459-777
fig|83333.1.peg.7 Putative alanine/glycine transport
protein 83333.1:NC_000913_7959-1431
fig|83333.1.peg.8 Transaldolase (EC 2.2.1.2)
83333.1:NC_000913_8238+954
```

If you compare the two outputs briefly, it becomes apparent that regions with hits of 5 or less represent spurious hits. It is clear that the default parameters are producing matches that are not real.

You have several parameters that determine the behavior of the "kmer searches":

1. The most obvious is the size of the kmers. In protein sequences we use a default size of 8 amino acids. An 8-mer is a "signature", if it has only been seen in proteins with a given function. In DNA searches, if we use kmers of length 8, we search for 24-character sequences that translate to a "signature kmer". Looking at the short piece of output relating to E.coli, it is clear that using kmers of length 8 is (in that case) leading to a situation in which there is a low-level (say, 1%) of the hits that are "spurious" in the sense that the matches are almost certainly not instances of the specified function (which is usually "hypothetical protein".

2. A minimum number of hits (occurrences of a "signature kmer" within the region). In the short piece of output displayed above, we were using a minimum number of hits of 3. It is clear from the output that, with kmers of size 8, we were getting hits of as many as five due to random occurrences within the DNA strings.

3. A "maxGap" parameter requires that the gap between occurrences of "signature kmers" be less than or equal to the specified value for the hits to be grouped into a single region. We were using a length of 600, which is probably too long.

4. Finally, we set a minimum size for a detected region. Note that you could recognize 5 occurrences of 8-mers within a DNA sequence of length 28, which might or might not be desirable.

I reran the experiment using kmers of length 9, a minimum number of hits of 3, a maximum gap of 300, and a minimum size of 48. I did this by running

```
    svr_assign_to_dna_using_figfams -by_location -kmer 9 -
maxGap 300 -minSize 48 -reliability 3 < ecoli.contigs >
better
```

This produced the following lines for the corresponding section of the E.coli genome:

```
NC_000913 13 NC_000913_190_253 Thr operon leader peptide
Escherichia coli K12
NC_000913 797 NC_000913_248_2797 Aspartokinase (EC 2.7.2.4)
/ Homoserine dehydrogenase (EC 1.1.1.3) Escherichia coli
K12
NC_000913 290 NC_000913_2801_3731 Homoserine kinase (EC
2.7.1.39) Escherichia coli K12
NC_000913 407 NC_000913_3734_5018 Threonine synthase (EC
4.2.3.1) Escherichia coli K12
NC_000913 122 NC_000913_5088_5674 hypothetical protein
Escherichia coli K12
NC_000913 76 NC_000913_5666_5312 hypothetical protein
Escherichia coli K12
NC_000913 238 NC_000913_6459_5685 UPF0246 protein YaaA
Escherichia coli K12
NC_000913 425 NC_000913_7959_6531 Putative alanine/glycine
transport protein Escherichia coli K12
NC_000913 310 NC_000913_8178_9189 Transaldolase (EC
2.2.1.2) Escherichia coli K12
```

This is substantially better. There is one bad call, but I suspect that it reflects actual annotations in other close E.coli genomes.

Now let us see whether the differences in parameters effect our ability to estimate populations in a real metagenomic sample. To see, I ran

```
svr_assign_to_dna_using_figfams -kmer 8 -maxGap 600 -
minSize 30 -reliability 3 < MG.sample |
svr_summarize_MG_output > function.summary.loose  2>
otu.summary.loose
```

and

```
svr_assign_to_dna_using_figfams -kmer 9 -maxGap 300 -
minSize 54 -reliability 3 < MG.sample |
svr_summarize_MG_output > function.summary.tighter  2>
otu.summary.tighter
```

The differences in estimates of population (i.e., a tabulation of the counts of hits against distinct OTUs) was

loose:

```
1714 0.110860 Staphylococcus aureus subsp. aureus COL
780 0.050450 Pseudomonas fluorescens SBW25
568 0.036738 Acinetobacter baumannii AB307-0294
510 0.032986 Acidovorax avenae subsp. citrulli AAC00-1
473 0.030593 Natranaerobius thermophilus JW/NM-WN-LF
443 0.028653 Streptococcus pneumoniae TIGR4
412 0.026648 Escherichia coli K12
351 0.022702 Stackebrandtia nassauensis DSM 44728
331 0.021409 Verminephrobacter eiseniae EF01-2
287 0.018563 Burkholderia cepacia R1808
269 0.017399 Xanthomonas campestris pv. campestris ATCC
33913
268 0.017334 Propionibacterium acnes KPA171202
234 0.015135 Delftia acidovorans SPH-1
231 0.014941 Synechococcus sp. WH 8102
227 0.014682 Serratia marcescens Db11
222 0.014359 Sphingomonas wittichii RW1
204 0.013194 Burkholderia fungorum
193 0.012483 Bradyrhizobium japonicum USDA 110
188 0.012160 Bacillus anthracis str. 'Ames Ancestor'
165 0.010672 Bordetella avium 197N
162 0.010478 Sulfurospirillum deleyianum DSM 6946
```

tighter:

```
1251 0.242301 Staphylococcus aureus subsp. aureus COL
442 0.085609 Pseudomonas fluorescens SBW25
329 0.063723 Streptococcus pneumoniae TIGR4
325 0.062948 Acidovorax avenae subsp. citrulli AAC00-1
319 0.061786 Acinetobacter baumannii AB307-0294
232 0.044935 Propionibacterium acnes KPA171202
227 0.043967 Verminephrobacter eiseniae EF01-2
196 0.037962 Escherichia coli K12
158 0.030602 Serratia marcescens Db11
130 0.025179 Delftia acidovorans SPH-1
107 0.020724 Herminiimonas arsenicoxydans
69 0.013364 Polaromonas sp. JS666
```

I have included counts of the OTUs that registered at least 1% of the total tabulated hits.

What does this mean?

1. I suppose that the obvious lessons are as follows:

2. I am writing this short note largely to illustrate the power of the server scripts – I think that it should be clear that a user can learn a number of important lessons without writing custom Perl code.

3. If one wished to do a serious analysis of a metagenomic sample using our simple tools, I would encourage them to "calibrate" the tools on data they have already characterized.

4. Using loose parameters to extract weak hits from lower-quality sequence should be viewed critically.

It seems to me that one might consider a 2-stage approach in which the population is first estimated fairly conservatively (pulling out the clear matches), and then a second less stringent pass could be made using the estimated population to "weight the results" (i.e., weak hits against OTUs that were shown to be present by the first pass would be taken more seriously in the second pass).

In any event, I hope that these comments make the tools somewhat more useful.

Appendix

The SEED Team

FIG

- Ross Overbeek
- Veronika Vonstein
- Sveta Gerdes
- Gordon Pusch
- Bruce Parrello
- Margo VanOeffelen
- Natalia Pilipenko
- Olga Vasieva, University of Liverpool
- Irina Goltsman
- Andrei Osterman, Burnham Institute
- Olga Zagnitko

ANL/UofC

- Rick Stevens
- Terry Disz
- Bob Olson
- Chris Henry
- Scott Devoid
- FangFang Xia
- Tobi Paczian
- Daniela Bartels

UIUC

- Gary Olson
- Jim Davis
- Claudia Reich

Hope College

- Matt DeJongh (Computer Science)
- Aaron Best (Biology)
- Nathan Tintle (Statistics)
- Hope College students

SDSU

- Ramy Aziz
- Rob Edwards, SDSU

SVR Routines

Annotation and Assertion Data Methods

svr_ach_lookup: Find protein assertions from the Annotation Clearinghouse.
svr_function_of: Get functions of protein-encoding genes
svr_assign_using_figfams: Assign Using the FIGfams Server
svr_protein_assertions: Get a list of Annotation Clearinghouse assertions for the specified proteins.

Annotation Support

svr_call_pegs: Call Genes using Annotation server.
svr_call_rnas: Call RNAs using Annotation server.
svr_metabolic_reconstruction: Get a metabolic reconstruction from a set of functional roles.
svr_assign_to_dna_using_figfams: Assign Using the FIGfams Server
svr_compare_feature_tables: usage: svr_compare_feature_tables old_features.tab new_fatures.tab [summary.yaml] > comparison.tab 2> summary.txt
svr_determine_sets_of_related_contigs: This takes as input a list of contig IDs. What is the output?
svr_role_to_pegs: Get PEGs that implement a given set of functional roles
svr_roles_to_subsys: Extend a set of roles to include the subsystems and category data

DNA and Protein Sequence Methods

svr_closest_genes: Locate genes in a specified genome containing the specified protein or DNA sequence.
svr_dna_seq: Produce DNA strings for contigs, FIG feature IDs, and/or locations.
svr_fasta: Produce DNA or protein strings for genes.
svr_upstream: Retrieve upstream regions from the Sapling Server.
svr_big_repeats: Find regions that appear to be big repeats (at the DNA level).
svr_cut_domain: Clip domains out of a set of protein sequences
svr_just_ends: Clip off the ends of a set of contigs
svr_possible_joins: Given kmer hits on ends of contigs, just group the hits having the same functions to support finding cases in which genes might span contigs.

Expression Data Methods

svr_coregulated_by_correspondence: Get genes that have evidence of coexpression indirectly (i.e., it seems to exist between corresponding genes in one or more other genomes with expression data).
svr_corr_by_exp: Get genes that have similar expression profiles.
svr_exp_genomes: List the names and IDs of all the genomes with expression data.

svr_find_regulatory_proteins: Find potential regulatory proteins

Feature (Gene) Data Methods

svr_aliases_of: Return all identifiers for genes in the database that are protein-sequence-equivalent to the specified identifiers. In this case, the identifiers are assumed to be in their natural form (without prefixes). For each identifier, the identified protein sequences will be found and then for each protein sequence, all identifiers for that protein sequence or for genes that produce that protein sequence will be returned.
svr_aliases_to_pegs: Convert aliases to PEGs
svr_evidence: Get evidence codes for protein-encoding genes
svr_gene_data: Get one or more pieces of data about each specified gene.
svr_neighbors_of: Get neighbors of protein-encoding genes (PEGs)
svr_similar_to: Get similarities for a PEG
svr_translations_of: Get translations from ids
svr_in_runs: Make sequences of genes into operons.

FIGfam Data Methods

svr_all_figfams: List all the features in each FIGfam.
svr_fc_figfams: Output the functionally coupled FIGfams By specifying a MinSc, you restrict the output to functionally-coupled FIGfams that co-occur in at least n OTUs.
svr_figfam_fasta: Produce FASTA strings for FIGfams.
svr_figfam_functions: Output the functions for the specified FIGfams.
svr_figfams_to_ids: List the PEGs for each specified FIGfam ID on STDOUT.
svr_assign_using_figfams: Assign Using the FIGfams Server
svr_ids_to_figfams: List the FIGfams for each specified gene ID on STDOUT. List on STDERR those lines where the id does not have a FIGFam.

Functional Coupling Data Methods

svr_cluster_pegs: Cluster PEGs that are close on the contig
svr_co_occurrence_evidence: Displays instances in which homologs of the two specified PEGs co-occur or members of two distinct FIGfams tend to co-occur. Thus you can say
svr_fc_figfams: Output the functionally coupled FIGfams By specifying a MinSc, you restrict the output to functionally-coupled FIGfams that co-occur in at least n OTUs.
svr_functionally_coupled: Get functionally_coupled neighbors (neighbors that tend to co-occur).

Genome Data Methods

svr_all_features: Get a list of Feature IDs for all features of a given type in a given genome or a list of genomes.
svr_all_genomes: List the names and IDs of all the (complete) genomes.
svr_close_genomes: List the IDs of the genomes that are functionally close to the input genomes.
svr_contigs_in_genome: For each incoming genome ID, return the IDs of its contigs.
svr_genome_functions: List the location and functional assignment for each gene in a specified genome.

svr_genome_statistics: Get one or more pieces of data about each specified genome.
svr_inherit_aliases: Cause a new genome to inherit aliases from an existing genome for protein-encoding genes that are unique within each genome and that have identical translations.
svr_inherit_annotations: Cause a new genome to inherit annotations from an existing genome for protein-encoding genes that are unique within each genome and that have identical translations.
svr_mapped_genomes: Get maps between a reference genome and a set of genomes to which you wish to compare the reference genome.
svr_members_of_otu: For each incoming genome ID, return the genome ID and name of each genome in the same organism taxonomic unit.
svr_otus: List the names and IDs of all the representative genomes for the organism taxonomic units in the system.
svr_taxonomy: Get taxonomy of genomes
svr_taxonomically_related_genomes: Get a list of genomes that are taxonomically related to the input genomes.
svr_distinct_otus: Classify the incoming genome IDs into organism taxonomic units.
svr_intergenic_regions: List all the intergenic regions in the contigs for a specified genome.
svr_discriminating_functions: Analyze two groups of genomes and return a list of the functions that discriminate between them.
svr_summarize_contigs: For each incoming genome ID, return statistics about its contigs.

Subsystem Data Methods

svr_all_subsystems: Get all the subsystem names.
svr_ids_to_subsystems: List the subsystems for each specified gene ID STDOUT. List on STDERR those lines where the id does not have a subsystem.
svr_pegs_in_subsystems: Return all genes in one or more subsystems found in one or more genomes.
svr_subsystem_genome_data: Output the features, variants, and roles for one or more subsystems, optionally filtered by genome ID.
svr_subsystem_genomes: Output the genomes of a subsystem.
svr_subsystem_roles: Output the roles of a subsystem.
svr_subsystem_spreadsheet: Output a subsystem's spreadsheet.

Alignment/Tree Methods

svr_align_seqs: This script takes a FASTA file from the standard input, aligns the sequences using Clustal, MUSCLE or MAFFT, and writes the alignment in the FASTA format to the standard output.
svr_all_models: List the existing metabolic models (and the genomes for which they were built).
svr_all_reactions: List the reactions IDs
svr_cohesion_groups: This script classifies tips of a newick tree into cohesion groups based on bootstrap values of tree branches.
svr_insert_seqs_into_alignment: This script takes a FASTA file of protein/DNA alignment from the standard input and the name of a FASTA file of sequences to be inserted from the command line and writes the resulting alignment to the standard output. When not possible, a message will be written to the standard error output.

svr_oligomer_similarity: This command goes through an alignment and computes the pairwise fractions of n-character identities, producing a matrix of values for each string length specified in the -min to -max parameters.
svr_reroot_tree: Reroot a tree at a different node or a point on an internal arc.
svr_sketch_tree: This little utility invokes a tree "printing" utility Gary Olsen wrote. It has a rich set of options. We suggest a default usage of -m and -a. Thus,
svr_tree: This script uses fasttree, PhyML or RAxML to build a maximum-likelihood tree from a FASTA alignment, or evaluates the likelihood of an input tree against a given alignment.
svr_tree_to_html: This script converts a newick tree into an HTML page. It has a rich set of options.
svr_trim_ali: This script takes a FASTA file of aligned sequences, trims the alignment by running PSIBLAST against the sequences themselves, and writes the trimmed alignment to the standard output.
svr_representative_sequences: usage: representative_sequences [opts] [rep_seqs_0] < new_seqs > rep_seqs

Chemistry Methods

svr_all_compounds: svr_all_compounds [options] [< modelIDFile] > output
svr_coupled_reactions: Takes as input a table containing reaction IDs and adds a column giving the "adjacent" reactions.
svr_get_compound_data: svr_get_compound_data [options] [< compoundIdFile] > output
svr_get_model_data: svr_get_model_data [options] [< modelIdFile] > output
svr_get_reaction_data: svr_get_reaction_data [options] [< reactionIdFile] > output
svr_reaction_description: This simple utility gives the reaction associated with reaction IDs.
svr_reactions_in_model: Takes as input a table containing model IDs and adds a column giving a reaction in the model. Since each model contains hundreds of reactions, the output file will be extremely large compared to the input file.
svr_reactions_to_roles: Takes as input a table containing reaction IDs and adds a column giving the roles that implement the reactions
svr_retreive_model: Retrieves reaction data for the selected metabolic model
svr_neighboring_reactions: Takes as input a table containing reaction IDs and adds 2 columns giving the distance and the connected reaction.
svr_roles_to_reactions: Extend a set of roles to include the associated reactions

Gap Filling Support (finding missing genes)

svr_find_clusters_relevant_to_reaction: Find clusters potentially relevant to a search for a "missing gene"
svr_find_hypos_for_cluster: Get candidates for a specific role by finding genes with no real assignment of function yet that are connected to a cluster. We will consider a hypothetical "connected to a cluster" iff
svr_gap_filled_reactions_and_roles: Get the reactions and functional roles that were predicted by gap-filling
svr_missing_roles: It is assumed that -r is used to specify a column in the input file. The column should contain reaction IDs for which "missing roles" might exist. The -g argument is used to specify the column containing the genome ID.
svr_neighborhood_of_role: Find roles in metabolic-function neighborhood

Utility Methods

svr_NCBI_taxonomy: Get taxonomy information from NCBI
svr_blast: Run blast locally
svr_psiblast_search: This script takes a FASTA file of trimmed protein sequence alignment, uses PSIBLAST to search against the protein database of complete genomes, and writes the extracted regions of hits to the standard output.
svr_add_lengths_to_blast: Add query and contig lengths to m8 blast output
svr_by_taxonomy: Separate a list by taxonomy
svr_file_to_spreadsheet: Writes the contents of the tab separated file on STDIN to a spreadsheet
svr_spreadsheet_to_file: Writes the contents of the spreadsheet given in filename to a tab separated file on STDOUT.
svr_sims2html: Build an HTML page or table from one or more tables of pairwise similarities.
svr_seed_to_table: Extract a tab-separated feature table from a SEED Genome Directory. Table format is:
svr_make_pan_genome_prot_families: Construct the protein families needed to study Pan Genomes
svr_summarize_MG_output: This simple program produces two summaries: one of the functions identified and one of the OTUs identified. We represent OTUs with a representative organism. The function summary is sent to stdout, while the OTU summary is sent to stderr.
svr_summarize_protein_families: Write out three simple reports relating to a proposed set of protein families.

Annotated list of Getting Started, Tutorial and Coding Examples

Getting Started	Links
A page of links exploring the recent RAST tutorials	RAST Tutorial Links
An introduction to the Fellowship for the Interpretation of Genomes, creators of the SEED	What is FIG
How to get a RAST account and access it to set your password	Video Tutorial: Creating a RAST Account
How to download, install and try out the SEED servers client	Accessing the SEED Servers:

code	Getting Started

Using the "svr" command line scripts	Links
Using svr command line scripts to retrieve all the features for a given genome. This is often the first step in an analysis sequence.	Find all features for a genome.
Using the svr command-line scripts to download all the genes for a genome.	Downloading a Genome
Using svr command-line scripts to get the roles and features of a subsystem.	Downloading a Subsystem
Using the svr command-line scripts to access the complete set of FIGfams.	Downloading the FIGfams
Using the svr command line scripts to access the Annotation Clearinghouse. How to see all of the assignments made by any group to a protein having the same sequence.	Getting all IDs, Aliases and Assertions of Function for One or more Protein sequences
Using svr command line scripts to find PEGs that are in the neighborhood of a given PEG.	Find Neighbors
Using svr command line scripts to see all the aliases by which a given feature or set of features is known in the SEED.	Find Gene Aliases
Using svr command line scripts to retrieve the assigned function for each of a set of genes.	Find Gene Function
Using svr command line scripts to retrieve all the features for a given genome. This is often the first step in an analysis sequence.	Find all features for a genome.
Two methods of doing annotations with the SEED	Annotating a genome using

servers 1. How to use svr command line scripts to do genome annotation: call the RNA-encoding genes and protein-encoding genes, asserts functions for the gene products, and produces an initial metabolic reconstruction. 2. How to use the Server Perl API to call the RNAs, PEGs, identify the functions, and generate an initial metabolic reconstruction.	the SEED servers (via the command line or Perl code)

Using the Server Perl API	Links
Using the Server Perl API to: Identify Genes Assign Functions to Encoded Proteins Create a Metabolic Reconstruction	Services to Support Annotation of Genes
Discussion of the example (Example 4) that uses the SEED servers Perl API to access to the data used to compute co-occurrence scores.	Access to Functional Coupling (Conserved Contiguity) Data
Discussion of server Perl API Example 3 illustrating the functions required to determine the location of a SEED gene encoding a specific protein and to acquire the genes from a given region centered on that location.	Creating Custom Interfaces
Using the server Perl API to identify the subsystems can be inferred from a set of functional roles.	Metabolic Reconstructions Provided for Complete Prokaryotic Genomes
Using the server Perl API to illustrate basic capabilities that	Conversion of Gene and

relate to determining the set of IDs attached to specific protein sequences.	Protein IDs

Metagenomics	Links
Using the RAST servers for Metagenomics	Getting Summaries of Functional Content and OTUs for an Metagenomic Sample

Publications describing SEED, RAST, and NMPDR tools

March 2011

SEED and RAST

Connecting Genotype to Phenotype in the Era of High-throughput Sequencing.
Henry CS, Overbeek R, Xia F, Best AA, Glass E, Gilbert J, Larsen P, Edwards R, Disz T, Meyer F, Vonstein V, Dejongh M, Bartels D, Desai N, D'Souza M, Devoid S, Keegan KP, Olson R, Wilke A, Wilkening J, Stevens RL.
Biochim Biophys Acta. 2011 Mar 18. [Epub ahead of print]
PMID: 21421023

High-throughput generation, optimization and analysis of genome-scale metabolic models.
Henry CS, DeJongh M, Best AA, Frybarger PM, Linsay B, Stevens RL.
Nat Biotechnol. 2010 Sep;28(9):977-82. Epub 2010 Aug 29.
PMID: 20802497

Accessing the SEED genome databases via Web services API: tools for programmers.
Disz T, Akhter S, Cuevas D, Olson R, Overbeek R, Vonstein V, Stevens R, Edwards RA.

BMC Bioinformatics. 2010 Jun 14;11:319.
PMID: 20546611

FIGfams: yet another set of protein families.
Meyer F, Overbeek R, Rodriguez A.
Nucleic Acids Res. 2009 Nov;37(20):6643-54. Epub 2009 Sep 17.
PMID: 19762480

The RAST Server: rapid annotations using subsystems technology.
Aziz RK, Bartels D, Best AA, DeJongh M, Disz T, Edwards RA, Formsma K, Gerdes S, Glass EM, Kubal M, Meyer F, Olsen GJ, Olson R, Osterman AL, Overbeek RA, McNeil LK, Paarmann D, Paczian T, Parrello B, Pusch GD, Reich C, Stevens R, Vassieva O, Vonstein V, Wilke A, Zagnitko O.
BMC Genomics. 2008 Feb 8;9:75.
PMID: 18261238

Annotation of bacterial and archaeal genomes: improving accuracy and consistency.
Overbeek R, Bartels D, Vonstein V, Meyer F.
Chem Rev. 2007 Aug;107(8):3431-47. Epub 2007 Jul 21. Review. No abstract available.
PMID: 17658903

Toward the automated generation of genome-scale metabolic networks in the SEED. FIG DeJongh M, Formsma K, Boillot P, Gould J, Rycenga M, Best A.BMC Bioinformatics. 2007 Apr 26;8:139.
PMID: 17462086

The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes.
Overbeek R, Begley T, Butler RM, Choudhuri JV, Chuang HY, Cohoon M, de CrŽcy-Lagard V, Diaz N, Disz T, Edwards R, Fonstein M, Frank ED, Gerdes S, Glass EM, Goesmann A, Hanson A, Iwata-Reuyl D, Jensen R, Jamshidi N, Krause L, Kubal M, Larsen N, Linke B, McHardy AC, Meyer F, Neuweger H, Olsen G, Olson R, Osterman A, Portnoy V, Pusch GD, Rodionov DA, RŸckert C, Steiner J, Stevens R, Thiele I, Vassieva O, Ye Y, Zagnitko O, Vonstein V.
Nucleic Acids Res. 2005 Oct 7;33(17):5691-702. Print 2005.
PMID: 16214803

NMPDR

The National Microbial Pathogen Database Resource (NMPDR): a genomics platform based on subsystem annotation.
McNeil LK, Reich C, Aziz RK, Bartels D, Cohoon M, Disz T, Edwards

RA, Gerdes S, Hwang K, Kubal M, Margaryan GR, Meyer F, Mihalo W, Olsen GJ, Olson R, Osterman A, Paarmann D, Paczian T, Parrello B, Pusch GD, Rodionov DA, Shi X, Vassieva O, Vonstein V, Zagnitko O, Xia F, Zinner J, Overbeek R, Stevens R.
Nucleic Acids Res. 2007 Jan;35(Database issue):D347-53.
PMID: 17145713

NIAID National Institute of Allergy and Infectious Diseases bioinformatics resource centers: New assets for pathogen informatics. Greene JM, Collins F, Lefkowitz EJ, Roos D, Scheuermann RH, Sobral B, Stevens R, White O, Di Francesco V.
Infect Immun. 2007 Jul;75(7):3212-9.
PMID: 17420237

Publications using SEED, RAST, or NMPDR tools

2011

Metabolic reconstruction of the archaeon methanogen Methanosarcina Acetivorans.
Satish Kumar V, Ferry JG, Maranas CD.
BMC Syst Biol. 2011 Feb 15;5:28.
PMID: 21324125

A role for the universal Kae1/Qri7/YgjD (COG0533) family in tRNA modification.
El Yacoubi B, Hatin I, Deutsch C, Kahveci T, Rousset JP, Iwata-Reuyl D, G Murzin A, de CrŽcy-Lagard V.
EMBO J. 2011 Mar 2;30(5):882-93. Epub 2011 Feb 1.
PMID: 21285948

Characterizing the native codon usages of a genome: an axis projection approach.
Davis JJ, Olsen GJ.
Mol Biol Evol. 2011 Jan;28(1):211-21. Epub 2010 Aug 2.
PMID: 20679093

2010

Building the blueprint of life.
Henry C, Overbeek R, Stevens RL.
Biotechnol J. 2010 Jul;5(7):695-704. Review.
PMID: 20665643

Towards a Systems Approach in the Genetic Analysis of Archaea: Accelerating Mutant Construction and Phenotypic Analysis in Haloferax volcanii
Ian K. Blaby, Gabriela Phillips, Crysten E. Blaby-Haas, Kevin S. Gulig, Basma El Yacoubi, and ValŽrie de CrŽcy-Lagard
Archaea. 2010 Dec 23;2010:426239.
PMID: 21234384

Discovery and characterization of an amidinotransferase involved in the modification of archaeal tRNA.
Phillips G, Chikwana VM, Maxwell A, El-Yacoubi B, Swairjo MA, Iwata-Reuyl D, de CrŽcy-Lagard V.
J Biol Chem. 2010 Apr 23;285(17):12706-13. Epub 2010 Feb 3.
PMID: 20129918

Two genome sequences of the same bacterial strain, Gluconacetobacter diazotrophicus PAl 5, suggest a new standard in genome sequence submission.
Giongo A, Tyler HL, Zipperer UN, Triplett EW.
Stand Genomic Sci. 2010 Jun 15;2(3):309-17.
PMID: 21304715

Functional Promiscuity of Homologues of the Bacterial ArsA ATPases.
Castillo R, Saier MH.
Int J Microbiol. 2010;2010:187373. Epub 2010 Oct 20.
PMID: 20981284

Comparative genomics of Gardnerella vaginalis strains reveals substantial differences in metabolic and virulence potential.
Yeoman CJ, Yildirim S, Thomas SM, Durkin AS, Torralba M, Sutton G, Buhay CJ, Ding Y, Dugan-Rocha SP, Muzny DM, Qin X, Gibbs RA, Leigh SR, Stumpf R, White BA, Highlander SK, Nelson KE, Wilson BA.
PLoS One. 2010 Aug 26;5(8):e12411.
PMID: 20865041

Microevolution of group A streptococci in vivo: capturing regulatory networks engaged in sociomicrobiology, niche adaptation, and hypervirulence.
Aziz RK, Kansal R, Aronow BJ, Taylor WL, Rowe SL, Kubal M, Chhatwal GS, Walker MJ, Kotb M.
PLoS One. 2010 Apr 14;5(4):e9798.
PMID: 20418946

Modal codon usage: assessing the typical codon usage of a genome.
Davis JJ, Olsen GJ.
Mol Biol Evol. 2010 Apr;27(4):800-10. Epub 2009 Dec 17.
PMID: 20018979

Reconstruction of xylose utilization pathway and regulons in Firmicutes.
Gu Y, Ding Y, Ren C, Sun Z, Rodionov DA, Zhang W, Yang S, Yang C, Jiang W.
BMC Genomics. 2010 Apr 21;11:255.
PMID: 20406496

Brucella abortus ure2 region contains an acid-activated urea transporter and a nickel transport system.
Sangari FJ, Cay—n AM, Seoane A, Garc'a-Lobo JM.
BMC Microbiol. 2010 Apr 10;10:107.
PMID: 20380737

Transposases are the most abundant, most ubiquitous genes in nature.
Aziz RK, Breitbart M, Edwards RA.
Nucleic Acids Res. 2010 Jul;38(13):4207-17. Epub 2010 Mar 9. Review.
PMID: 20215432

Predicting the pathway involved in post-translational modification of elongation factor P in a subset of bacterial species.
Bailly M, de CrŽcy-Lagard V.
Biol Direct. 2010 Jan 13;5:3.
PMID: 20070887

The novel polysaccharide deacetylase homologue Pdi contributes to virulence of the aquatic pathogen Streptococcus iniae.
Milani CJ, Aziz RK, Locke JB, Dahesh S, Nizet V, Buchanan JT.
Microbiology. 2010 Feb;156(Pt 2):543-54. Epub 2009 Sep 17.
PMID: 1976244

Network analyses structure genetic diversity in independent genetic worlds.
Halary S, Leigh JW, Cheaib B, Lopez P, Bapteste E.
Proc Natl Acad Sci U S A. 2010 Jan 5;107(1):127-32. Epub 2009 Dec 10.
PMID: 20007769

FolX and FolM are essential for tetrahydromonapterin synthesis in

Escherichia coli and Pseudomonas aeruginosa.
Pribat A, Blaby IK, Lara-Nœ—ez A, Gregory JF 3rd, de CrŽcy-Lagard V, Hanson AD.
J Bacteriol. 2010 Jan;192(2):475-82. Epub 2009 Nov 6.
PMID: 19897652

Biosynthesis of wyosine derivatives in tRNA: an ancient and highly diverse pathway in Archaea.
de CrŽcy-Lagard V, Brochier-Armanet C, Urbonavicius J, Fernandez B, Phillips G, Lyons B, Noma A, Alvarez S, Droogmans L, Armengaud J, Grosjean H.
Mol Biol Evol. 2010 Sep;27(9):2062-77. Epub 2010 Apr 9
PMID: 20382657

Genomics-driven reconstruction of acinetobacter NAD metabolism: insights for antibacterial target selection.
Sorci L, Blaby I, De Ingeniis J, Gerdes S, Raffaelli N, de CrŽcy Lagard V, Osterman A.
J Biol Chem. 2010 Dec 10;285(50):39490-9. Epub 2010 Oct 6.
PMID: 20926389

Moonlighting glutamate formiminotransferases can functionally replace 5-formyltetrahydrofolate cycloligase.
Jeanguenin L, Lara-Nœ—ez A, Pribat A, Mageroy MH, Gregory JF 3rd, Rice KC, de CrŽcy-Lagard V, Hanson AD.
J Biol Chem. 2010 Dec 31;285(53):41557-66. Epub 2010 Oct 15.
PMID: 20952389

A role for tetrahydrofolates in the metabolism of iron-sulfur clusters in all domains of life.
Waller JC, Alvarez S, Naponelli V, Lara-Nu—ez A, Blaby IK, Da Silva V, Ziemak MJ, Vickers TJ, Beverley SM, Edison AS, Rocca JR, Gregory JF 3rd, de CrŽcy-Lagard V, Hanson AD.
Proc Natl Acad Sci U S A. 2010 Jun 8;107(23):10412-7. Epub 2010 May 20.
PMID: 20489182

Genomic encyclopedia of sugar utilization pathways in the Shewanella genus.
Rodionov DA, Yang C, Li X, Rodionova IA, Wang Y, Obraztsova AY, Zagnitko OP, Overbeek R, Romine MF, Reed S, Fredrickson JK, Nealson KH, Osterman AL.
BMC Genomics. 2010 Sep 13;11:494.
PMID: 20836887

Viral and microbial community dynamics in four aquatic

environments.
Rodriguez-Brito B, Li L, Wegley L, Furlan M, Angly F, Breitbart M, Buchanan J, Desnues C, Dinsdale E, Edwards R, Felts B, Haynes M, Liu H, Lipson D, Mahaffy J, Martin-Cuadrado AB, Mira A, Nulton J, Pasi? L, Rayhawk S, Rodriguez-Mueller J, Rodriguez-Valera F, Salamon P, Srinagesh S, Thingstad TF, Tran T, Thurber RV, Willner D, Youle M, Rohwer F.
ISME J. 2010 Jun;4(6):739-51.
PMID: 20147985

2009

iBsu1103: a new genome-scale metabolic model of Bacillus subtilis based on SEED annotations.
Henry CS, Zinner JF, Cohoon MP, Stevens RL.
Genome Biol. 2009;10(6):R69. Epub 2009 Jun 25.
PMID: 19555510

Cohesion group approach for evolutionary analysis of aspartokinase, an enzyme that feeds a branched network of many biochemical pathways.
Lo CC, Bonner CA, Xie G, D'Souza M, Jensen RA.
Microbiol Mol Biol Rev. 2009 Dec;73(4):594-651. Review.
PMID: 19946135

The universal YrdC/Sua5 family is required for the formation of threonylcarbamoyladenosine in tRNA.
El Yacoubi B, Lyons B, Cruz Y, Reddy R, Nordin B, Agnelli F, Williamson JR, Schimmel P, Swairjo MA, de CrŽcy-Lagard V.
Nucleic Acids Res. 2009 May;37(9):2894-909. Epub 2009 Mar 13.
PMID: 19287007

A subset of the diverse COG0523 family of putative metal chaperones is linked to zinc homeostasis in all kingdoms of life.
Haas CE, Rodionov DA, Kropat J, Malasarn D, Merchant SS, de CrŽcy-Lagard V.
BMC Genomics. 2009 Oct 12;10:470.
PMID: 19822009

Structure of PhnP, a phosphodiesterase of the carbon-phosphorus lyase pathway for phosphonate degradation.
Podzelinska K, He SM, Wathier M, Yakunin A, Proudfoot M, Hove-Jensen B, Zechel DL, Jia Z.
J Biol Chem. 2009 Jun 19;284(25):17216-26. Epub 2009 Apr 14.
PMID: 19366688

Adaptations to submarine hydrothermal environments exemplified by

the genome of Nautilia profundicola. RAST Campbell BJ, Smith JL, Hanson TE, Klotz MG, Stein LY, Lee CK, Wu D, Robinson JM, Khouri HM, Eisen JA, Cary SC. PLoS Genet. 2009 Feb;5(2):e1000362.
PMID: 19197347

Biogenesis and Homeostasis of Nicotinamide Adenine Dinucleotide Cofactor. FIG Osterman A. "Module 3.6.3.10" Posted February 13, 2009 in A. Bšck, R. Curtiss III, J. B. Kaper, P. D. Karp, F. C. Neidhardt, T. Nystršm, J. M. Slauch, and C. L. Squires, and D. Ussery (ed.), EcoSalÑ Escherichia coli and Salmonella: Cellular and molecular biology. http://www.ecosal.org. ASM Press, Washington, DC.

Three-dimensional structural view of the central metabolic network of Thermotoga maritima.
Zhang Y, Thiele I, Weekes D, Li Z, Jaroszewski L, Ginalski K, Deacon AM, Wooley J, Lesley SA, Wilson IA, Palsson B, Osterman A, Godzik A.
Science. 2009 Sep 18;325(5947):1544-9.
PMID: 19762644

Red death in Caenorhabditis elegans caused by Pseudomonas aeruginosa PAO1.
Zaborin A, Romanowski K, Gerdes S, Holbrook C, Lepine F, Long J, Poroyko V, Diggle SP, Wilke A, Righetti K, Morozova I, Babrowski T, Liu DC, Zaborina O, Alverdy JC.
Proc Natl Acad Sci U S A. 2009 Apr 14;106(15):6327-32. Epub 2009 Apr 6.
PMID: 19369215

Furanose-specific sugar transport: characterization of a bacterial galactofuranose-binding protein.
Horler RS, MŸller A, Williamson DC, Potts JR, Wilson KS, Thomas GH.
J Biol Chem. 2009 Nov 6;284(45):31156-63. Epub 2009 Sep 10.
PMID: 19744923

Microbial NAD metabolism: lessons from comparative genomics.
Gazzaniga F, Stebbins R, Chang SZ, McPeek MA, Brenner C.
Microbiol Mol Biol Rev. 2009 Sep;73(3):529-41, Table of Contents. Review.
PMID: 19721089

'Unknown' proteins and 'orphan' enzymes: the missing half of the engineering parts list--and how to find it.
Hanson AD, Pribat A, Waller JC, de CrŽcy-Lagard V.

Biochem J. 2009 Dec 14;425(1):1-11. Review.
PMID: 20001958

6-pyruvoyltetrahydropterin synthase paralogs replace the folate synthesis enzyme dihydroneopterin aldolase in diverse bacteria.
Pribat A, Jeanguenin L, Lara-Nœ—ez A, Ziemak MJ, Hyde JE, de CrŽcy-Lagard V, Hanson AD.
J Bacteriol. 2009 Jul;191(13):4158-65. Epub 2009 Apr 24.
PMID: 19395485

Comparative genomics of regulation of fatty acid and branched-chain amino acid utilization in proteobacteria.
Kazakov AE, Rodionov DA, Alm E, Arkin AP, Dubchak I, Gelfand MS.
J Bacteriol. 2009 Jan;191(1):52-64. Epub 2008 Sep 26.
PMID: 18820024

Nicotinamide mononucleotide synthetase is the key enzyme for an alternative route of NAD biosynthesis in Francisella tularensis.
Sorci L, Martynowski D, Rodionov DA, Eyobo Y, Zogaj X, Klose KE, Nikolaev EV, Magni G, Zhang H, Osterman AL.
Proc Natl Acad Sci U S A. 2009 Mar 3;106(9):3083-8. Epub 2009 Feb 9.
PMID: 19204287

Genomic reconstruction of Shewanella oneidensis MR-1 metabolism reveals a previously uncharacterized machinery for lactate utilization.
Pinchuk GE, Rodionov DA, Yang C, Li X, Osterman AL, Dervyn E, Geydebrekht OV, Reed SB, Romine MF, Collart FR, Scott JH, Fredrickson JK, Beliaev AS.
Proc Natl Acad Sci U S A. 2009 Feb 24;106(8):2874-9. Epub 2009 Feb 5.
PMID: 19196979

Metagenomic analysis of respiratory tract DNA viral communities in cystic fibrosis and non-cystic fibrosis individuals.
Willner D, Furlan M, Haynes M, Schmieder R, Angly FE, Silva J, Tammadoni S, Nosrat B, Conrad D, Rohwer F.
PLoS One. 2009 Oct 9;4(10):e7370.
PMID: 19816605

Role and regulation of fatty acid biosynthesis in the response of Shewanella piezotolerans WP3 to different temperatures and pressures.
Wang F, Xiao X, Ou HY, Gai Y, Wang F.
J Bacteriol. 2009 Apr;191(8):2574-84. Epub 2009 Feb 6.

PMID: 19201790

Characterization of the LacI-type transcriptional repressor RbsR controlling ribose transport in Corynebacterium glutamicum ATCC 13032. FIG Nentwich SS, Brinkrolf K, Gaigalat L, HŸser AT, Rey DA, Mohrbach T, Marin K, PŸhler A, Tauch A, Kalinowski J. Microbiology. 2009 Jan;155(1):150-64.
PMID: 19118356

A novel class of modular transporters for vitamins in prokaryotes.
Rodionov DA, Hebbeln P, Eudes A, ter Beek J, Rodionova IA, Erkens GB, Slotboom DJ, Gelfand MS, Osterman AL, Hanson AD, Eitinger T. J Bacteriol. 2009 Jan;191(1):42-51. Epub 2008 Oct 17.
PMID: 18931129

2008

The type III pantothenate kinase encoded by coaX is essential for growth of Bacillus anthracis.
Paige C, Reid SD, Hanna PC, Claiborne A.
J Bacteriol. 2008 Sep;190(18):6271-5. Epub 2008 Jul 18.
PMID: 18641144

Comparative genomics of two ecotypes of the marine planktonic copiotroph Alteromonas macleodii suggests alternative lifestyles associated with different kinds of particulate organic matter. FIG Ivars-Martinez E, Martin-Cuadrado AB, D'Auria G, Mira A, Ferriera S, Johnson J, Friedman R, Rodriguez-Valera F. ISME J. 2008 Dec;2(12):1194-212.
PMID: 18670397

Gene order phylogeny of the genus Prochlorococcus. FIG Luo H, Shi J, Arndt W, Tang J, Friedman R. PLoS ONE. 2008;3(12):e3837.
PMID: 19050756

The dual transcriptional regulator CysR in Corynebacterium glutamicum ATCC 13032 controls a subset of genes of the McbR regulon in response to the availability of sulphide acceptor molecules. FIG RŸckert C, Milse J, Albersmeier A, Koch DJ, PŸhler A, Kalinowski J. BMC Genomics. 2008 Oct 14;9:483.
PMID: 18854009

Biosynthesis of 7-deazaguanosine-modified tRNA nucleosides: a new role for GTP cyclohydrolase I.
Phillips G, El Yacoubi B, Lyons B, Alvarez S, Iwata-Reuyl D, de

CrŽcy-Lagard V.
J Bacteriol. 2008 Dec;190(24):7876-84. Epub 2008 Oct 17.
PMID: 18931107

Bifunctional NMN adenylyltransferase/ADP-ribose pyrophosphatase: structure and function in bacterial NAD metabolism.
Huang N, Sorci L, Zhang X, Brautigam CA, Li X, Raffaelli N, Magni G, Grishin NV, Osterman AL, Zhang H.
Structure. 2008 Feb;16(2):196-209.
PMID: 18275811

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 FIG Martin-Cuadrado AB, Rodriguez-Valera F, Moreira D, Alba JC, Ivars-Mart'nez E, Henn MR, Talla E, L—pez-Garc'a P. ISME J. 2008 Aug;2(8):865-86.
PMID: 18463691

Towards environmental systems biology of Shewanella. FIG Fredrickson JK, Romine MF, Beliaev AS, Auchtung JM, Driscoll ME, Gardner TS, Nealson KH, Osterman AL, Pinchuk G, Reed JL, Rodionov DA, Rodrigues JL, Saffarini DA, Serres MH, Spormann AM, Zhulin IB, Tiedje JM. Nat Rev Microbiol. 2008 Aug;6(8):592-603.
PMID: 18604222

Identification and characterization of genes underlying chitinolysis in Collimonas fungivorans Ter331. FIG Fritsche K, de Boer W, Gerards S, van den Berg M, van Veen JA, Leveau JH. FEMS Microbiol Ecol. 2008 Oct;66(1):123-35.
PMID: 18671744

Identification of a cellobiose utilization gene cluster with cryptic beta-galactosidase activity in Vibrio fischeri.
Adin DM, Visick KL, Stabb EV.
Appl Environ Microbiol. 2008 Jul;74(13):4059-69. Epub 2008 May 16.
PMID: 18487409

Rise and persistence of global M1T1 clone of Streptococcus pyogenes. FIG Aziz RK, Kotb M. Emerg Infect Dis. 2008 Oct;14(10):1511-7.
PMID: 18826812

Vibrio cholerae VciB promotes iron uptake via ferrous iron transporters. nmpdr Mey AR, Wyckoff EE, Hoover LA, Fisher CR,

Payne SM. J Bacteriol. 2008 Sep;190(17):5953-62.
PMID: 18586940

Plasmodium falciparum: a paradigm for alternative folate biosynthesis in diverse microorganisms?
Hyde JE, Dittrich S, Wang P, Sims PF, de CrŽcy-Lagard V, Hanson AD.
Trends Parasitol. 2008 Nov;24(11):502-8. Epub 2008 Sep 19.
PMID: 18805734

Phylogenomic and functional analysis of pterin-4a-carbinolamine dehydratase family (COG2154) proteins in plants and microorganisms.
Naponelli V, Noiriel A, Ziemak MJ, Beverley SM, Lye LF, Plume AM, Botella JR, Loizeau K, Ravanel S, RŽbeillŽ F, de CrŽcy-Lagard V, Hanson AD.
Plant Physiol. 2008 Apr;146(4):1515-27. Epub 2008 Feb 1.
PMID: 18245455

Plasmodium falciparum: a paradigm for alternative folate biosynthesis in diverse microorganisms?
Hyde JE, Dittrich S, Wang P, Sims PF, de CrŽcy-Lagard V, Hanson AD.
Trends Parasitol. 2008 Nov;24(11):502-8. Epub 2008 Sep 19.
PMID: 18805734

ComPath: Comparative enzyme analysis and annotation in pathway/subsystem contexts. FIG Choi K, Kim S. BMC Bioinformatics. 2008 Mar 6;9:145.
PMID: 18325116

Cohesion group approach for evolutionary analysis of TyrA, a protein family with wide-ranging substrate specificities.
Bonner CA, Disz T, Hwang K, Song J, Vonstein V, Overbeek R, Jensen RA.
Microbiol Mol Biol Rev. 2008 Mar;72(1):13-53, table of contents. Review.
PMID: 18322033

Critical evaluation of two primers commonly used for amplification of bacterial 16S rRNA genes.
Frank JA, Reich CI, Sharma S, Weisbaum JS, Wilson BA, Olsen GJ.
Appl Environ Microbiol. 2008 Apr;74(8):2461-70. Epub 2008 Feb 22.
PMID: 18296538

Sialic acid mutarotation is catalyzed by the Escherichia coli

beta-propeller protein YjhT. FIG Severi E, MŸller A, Potts JR, Leech A, Williamson D, Wilson KS, Thomas GH. J Biol Chem. 2008 Feb 22;283(8):4841-9.
PMID: 18063573

Gene set analyses for interpreting microarray experiments on prokaryotic organisms.
Tintle NL, Best AA, DeJongh M, Van Bruggen D, Heffron F, Porwollik S, Taylor RC.
BMC Bioinformatics. 2008 Nov 5;9:469.
PMID: 18986519

Biochemical and phylogenetic characterization of a novel diaminopimelate biosynthesis pathway in prokaryotes identifies a diverged form of LL-diaminopimelate aminotransferase.
Hudson AO, Gilvarg C, Leustek T.
J Bacteriol. 2008 May;190(9):3256-63. Epub 2008 Feb 29.
PMID: 18310350

Transcriptional regulation of NAD metabolism in bacteria: genomic reconstruction of NiaR (YrxA) regulon.
Rodionov DA, Li X, Rodionova IA, Yang C, Sorci L, Dervyn E, Martynowski D, Zhang H, Gelfand MS, Osterman AL.
Nucleic Acids Res. 2008 Apr;36(6):2032-46. Epub 2008 Feb 14.
PMID: 18276644

Transcriptional regulation of NAD metabolism in bacteria: NrtR family of Nudix-related regulators.
Rodionov DA, De Ingeniis J, Mancini C, Cimadamore F, Zhang H, Osterman AL, Raffaelli N.
Nucleic Acids Res. 2008 Apr;36(6):2047-59. Epub 2008 Feb 14.
PMID: 18276643

Glycerate 2-kinase of Thermotoga maritima and genomic reconstruction of related metabolic pathways.
Yang C, Rodionov DA, Rodionova IA, Li X, Osterman AL.
J Bacteriol. 2008 Mar;190(5):1773-82. Epub 2007 Dec 21.
PMID: 18156253

Comparative approach to analysis of gene essentiality.
Osterman AL, Gerdes SY.
Methods Mol Biol. 2008;416:459-66. No abstract available.
PMID: 18392987

Microbial ecology of four coral atolls in the Northern Line

Islands.
Dinsdale EA, Pantos O, Smriga S, Edwards RA, Angly F, Wegley L, Hatay M, Hall D, Brown E, Haynes M, Krause L, Sala E, Sandin SA, Thurber RV, Willis BL, Azam F, Knowlton N, Rohwer F.
PLoS One. 2008 Feb 27;3(2):e1584.
PMID: 18301735

2007

IUBMB Life. 2007 Oct;59(10):634-58.
Comparative RNomics and modomics in Mollicutes: prediction of gene function and evolutionary implications.
de CrŽcy-Lagard V, Marck C, Brochier-Armanet C, Grosjean H.
PMID: 17852564

Trends Microbiol. 2007 Dec;15(12):563-70. Epub 2007 Nov 9.
Finding novel metabolic genes through plant-prokaryote phylogenomics.
de CrŽcy-Lagard V, Hanson AD.
PMID: 17997099

An in vivo expression technology screen for Vibrio cholerae genes expressed in human volunteers. nmpdr Lombardo MJ, Michalski J, Martinez-Wilson H, Morin C, Hilton T, Osorio CG, Nataro JP, Tacket CO, Camilli A, Kaper JB. Proc Natl Acad Sci U S A. 2007 Nov 13;104(46):18229-34.
PMID: 17986616

The biological role of death and lysis in biofilm development.
Bayles KW. Nat Rev Microbiol. 2007 Sep;5(9):721-6.
PMID: 17694072

Whole proteome analysis of post-translational modifications: Applications of mass-spectrometry for proteogenomic annotation. FIG Gupta N, Tanner S, Jaitly N, Adkins JN, Lipton M, Edwards R, Romine M, Osterman A, Bafna V, Smith RD, Pevzner PA. Genome Res. 2007 Sep;17(9):1362-77.
PMID: 17690205

Finding novel metabolic genes through plant-prokaryote phylogenomics.
de CrŽcy-Lagard V, Hanson AD.
Trends Microbiol. 2007 Dec;15(12):563-70. Epub 2007 Nov 9. Review.
PMID: 17997099

Comparative genomics of bacterial and plant folate synthesis and

salvage: predictions and validations.
de CrŽcy-Lagard V, El Yacoubi B, de la Garza RD, Noiriel A, Hanson AD.
BMC Genomics. 2007 Jul 23;8:245.
PMID: 17645794

Characterization of a TIR-like protein from Paracoccus denitrificans. nmpdr Low LY, Mukasa T, Reed JC, Pascual J.
Biochem Biophys Res Commun. 2007 May 4;356(2):481-6.
PMID: 17362878

Comparative genomic reconstruction of transcriptional regulatory networks in bacteria.
Rodionov DA.
Chem Rev. 2007 Aug;107(8):3467-97. Epub 2007 Jul 18. Review. No abstract available.
PMID: 17636889

Free methionine-(R)-sulfoxide reductase from Escherichia coli reveals a new GAF domain function.
Lin Z, Johnson LC, Weissbach H, Brot N, Lively MO, Lowther WT.
Proc Natl Acad Sci U S A. 2007 Jun 5;104(23):9597-602. Epub 2007 May 29.
PMID: 17535911

Identification of genes encoding tRNA modification enzymes by comparative genomics.
de CrŽcy-Lagard V.
Methods Enzymol. 2007;425:153-83. Review.
PMID: 17673083

Structure of the type III pantothenate kinase from Bacillus anthracis at 2.0 A resolution: Implications for coenzyme A-dependent redox biology. FIG Nicely NI, Parsonage D, Paige C, Newton GL, Fahey RC, Leonardi R, Jackowski S, Mallett TC, Claiborne A. Biochemistry. 2007 Mar 20;46(11):3234-45.
PMID: 17323930

Biotin uptake in prokaryotes by solute transporters with an optional ATP-binding cassette-containing module.
Hebbeln P, Rodionov DA, Alfandega A, Eitinger T.
Proc Natl Acad Sci U S A. 2007 Feb 20;104(8):2909-14. Epub 2007 Feb 14.
PMID: 17301237

The IclR-type transcriptional repressor LtbR regulates the

expression of leucine and tryptophan biosynthesis genes in the amino acid producer Corynebacterium glutamicum.
Brune I, Jochmann N, Brinkrolf K, HŸser AT, Gerstmeir R, Eikmanns BJ, Kalinowski J, PŸhler A, Tauch A.
J Bacteriol. 2007 Apr;189(7):2720-33. Epub 2007 Jan 26.
PMID: 17259312

Genomic identification and in vitro reconstitution of a complete biosynthetic pathway for the osmolyte di-myo-inositol-phosphate.
Rodionov DA, Kurnasov OV, Stec B, Wang Y, Roberts MF, Osterman AL.
Proc Natl Acad Sci U S A. 2007 Mar 13;104(11):4279-84. Epub 2007 Mar 2.
PMID: 17360515

2006

El Yacoubi B, Bonnett S, Anderson JN, Swairjo MA, Iwata-Reuyl D, de CrŽcy-Lagard V. Discovery of a new prokaryotic type I GTP cyclohydrolase family.
J Biol Chem. 2006 Dec 8;281(49):37586-93. Epub 2006 Oct 10.
PMID: 17032654

A hidden metabolic pathway exposed.
Osterman A.
Proc Natl Acad Sci U S A. 2006 Apr 11;103(15):5637-8. Epub 2006 Apr 4. No abstract available.
PMID: 16595627

Comparative and functional genomic analysis of prokaryotic nickel and cobalt uptake transporters: evidence for a novel group of ATP-binding cassette transporters.
Rodionov DA, Hebbeln P, Gelfand MS, Eitinger T.
J Bacteriol. 2006 Jan;188(1):317-27.
PMID: 16352848

Crystal structure of a type III pantothenate kinase: insight into the mechanism of an essential coenzyme A biosynthetic enzyme universally distributed in bacteria.
Yang K, Eyobo Y, Brand LA, Martynowski D, Tomchick D, Strauss E, Zhang H.
J Bacteriol. 2006 Aug;188(15):5532-40.
PMID: 16855243

Essential genes on metabolic maps. FIG Gerdes S, Edwards R, Kubal M, Fonstein M, Stevens R, Osterman A. Curr Opin Biotechnol. 2006 Oct;17(5):448-56.

PMID: 16978855

Comparative genomics and experimental characterization of N-acetylglucosamine utilization pathway of Shewanella oneidensis.
Yang C, Rodionov DA, Li X, Laikova ON, Gelfand MS, Zagnitko OP, Romine MF, Obraztsova AY, Nealson KH, Osterman AL.
J Biol Chem. 2006 Oct 6;281(40):29872-85. Epub 2006 Jul 20.
PMID: 16857666

Comparative genomics of NAD biosynthesis in cyanobacteria.
Gerdes SY, Kurnasov OV, Shatalin K, Polanuyer B, Sloutsky R, Vonstein V, Overbeek R, Osterman AL.
J Bacteriol. 2006 Apr;188(8):3012-23.
PMID: 16585762

Experimental and computational assessment of conditionally essential genes in Escherichia coli.
Joyce AR, Reed JL, White A, Edwards R, Osterman A, Baba T, Mori H, Lesely SA, Palsson B¯, Agarwalla S.
J Bacteriol. 2006 Dec;188(23):8259-71. Epub 2006 Sep 29.
PMID: 17012394

Characterization of the Staphylococcus aureus heat shock, cold shock, stringent, and SOS responses and their effects on log-phase mRNA turnover.
Anderson KL, Roberts C, Disz T, Vonstein V, Hwang K, Overbeek R, Olson PD, Projan SJ, Dunman PM.
J Bacteriol. 2006 Oct;188(19):6739-56.
PMID: 16980476

Genome sequence of the bioplastic-producing "Knallgas" bacterium Ralstonia eutropha H16. FIG Pohlmann A, Fricke WF, Reinecke F, Kusian B, Liesegang H, Cramm R, Eitinger T, Ewering C, Pštter M, Schwartz E, Strittmatter A, Voss I, Gottschalk G, SteinbŸchel A, Friedrich B, Bowien B. Nat Biotechnol. 2006 Oct;24(10):1257-62.
PMID: 16964242

Roberts, C., K. L. Anderson, E. Murphy, S. J. Projan, W. Mounts, B. Hurlburt, M. Smeltzer, R. Overbeek, T. Disz, and P. M. Dunman. 2006. Characterizing the effect of the Staphylococcus aureus virulence factor regulator, SarA, on log-phase mRNA half-lives.
J. Bacteriol. 188:2593-2603.
PMID: 16547047

The thioredoxin domain of Neisseria gonorrhoeae PilB can use electrons from DsbD to reduce downstream methionine sulfoxide

reductases. FIG Brot N, Collet JF, Johnson LC, Jonsson TJ, Weissbach H, Lowther WT. J Biol Chem. 2006 Oct 27;281(43):32668-75.
PMID: 16926157

Random mutagenesis in Corynebacterium glutamicum ATCC 13032 using an IS6100-based transposon vector identified the last unknown gene in the histidine biosynthesis pathway. FIG Mormann S, Lšmker A, RŸckert C, Gaigalat L, Tauch A, PŸhler A, Kalinowski J. BMC Genomics. 2006 Aug 10;7:205.
PMID: 16901339

Study of an alternate glyoxylate cycle for acetate assimilation by Rhodobacter sphaeroides. FIG Alber BE, Spanheimer R, Ebenau-Jehle C, Fuchs G. Mol Microbiol. 2006 Jul;61(2):297-309.
PMID: 16856937

Community genomics among stratified microbial assemblages in the ocean's interior. FIG DeLong EF, Preston CM, Mincer T, Rich V, Hallam SJ, Frigaard NU, Martinez A, Sullivan MB, Edwards R, Brito BR, Chisholm SW, Karl DM. Science. 2006 Jan 27;311(5760):496-503.
PMID: 16439655

2005

Functional genomics and expression analysis of the Corynebacterium glutamicum fpr2-cysIXHDNYZ gene cluster involved in assimilatory sulphate reduction.
RŸckert C, Koch DJ, Rey DA, Albersmeier A, Mormann S, PŸhler A, Kalinowski J.
BMC Genomics. 2005 Sep 13;6:121.
PMID: 16159395

Low-molecular-weight protein tyrosine phosphatases of Bacillus subtilis. Musumeci L, Bongiorni C, Tautz L, Edwards RA, Osterman A, Perego M, Mustelin T, Bottini N. J Bacteriol. 2005 Jul;187(14):4945-56.
PMID: 15995210

Index

A

aliases, 45

Annotating a Genome Using RAST, 19

Annotating a Genome with myRAST, 25

Annotating a Prokaryotic Genome, 17

Annotation Support Server (API), 34

Annotator's SEED, 7

A-SEED, 7

B

biomass reaction, 15

C

Command Line "svr" scripts, 35

Command Line Services, 43

compare regions, 28

E

Entity-Relationship Model (of the SEED data), 33

Exporting Your Genome from myRAST, 29

Exporting Your Genome from RAST, 25

F

Fellowship for Interpretation of Genomes, 4

FIG, 4

FIGfam, 13

FIGfams, 4

Find all features for a genome, 44

Find Gene Aliases, 45

Find Gene Function, 44

Find Neighbors, 46

flux-balance, 31

functional role, 13

I

isofunctional homologs, 14

K

kmer searches, 119

M

manual annnotation, 14

metabolic model, 15, 31

Metabolic Reconstruction, 18

Metagenomic Sample, 116

Metagenomics Sample, 117

Model Server (API), 35

Modeling, 31

myRAST shell, 43

myRAST Shell, 35

P

P1k Project, 13

PATRIC SEED, 7

Pipe SVR scripts, 36

Project to Annotate 1000 Genomes, 13

P-SEED, 7

Public SEED, 7

PubSEED, 7

R

Rapid Annotation using Subsytems Technology, 4

Rapid Annotations using Subsystems Technology, 14

RAST, 4, 14

RAST Batch Interface, 47

RAST server (API), 34

S

Sapling DB, 33

Sapling Server (API), 34

SAS_SERVER, 43

SEED, 7

SEED Project, 4

Servers (Getting Started), 35

Servers (Summary), 34

Servers Blog, 38

shell scripts, 43

stoichiometric matrix, 15

Subsystem, 13

Subsystems Approach to Genome Annotation, 13

Subsytems, 4

svr scripts, 35

svr scripts (list of all), 35

svr_, 43

svr__submit_RAST_job, 47

svr_all_features, 36, 118

svr_all_genomes, 35

svr_assign_to_dna_using_figfams, 116, 117

svr_contigs_in_genome, 117

svr_delete_RAST_job, 49

svr_function_of, 36

svr_kill_RAST_job, 49

svr_metabolic_reconstruction, 117

svr_retrieve_RAST_job, 49

svr_run_RAST_jobs, 48

svr_status_of_RAST_job, 48

svr_summarize_MG_output, 116

T

The annotation cycle, 4

U

`use SeedEnv`, 37

W

walking your genome, 17

Walking your Genome, 20

Walking your Genome Using myRAST, 28

windows user, 43

writing Perl scripts to access the servers, 37