JOHNS HOPKINS UNIVERSITY

Zanvyl Krieger School of Arts and Sciences
Advanced Academic Programs

410.734.81 and 410.734.82
Practical Introduction to Metagenomics

Topic: Assembly and gene prediction

Instructor: Joshua Orvis

Overview

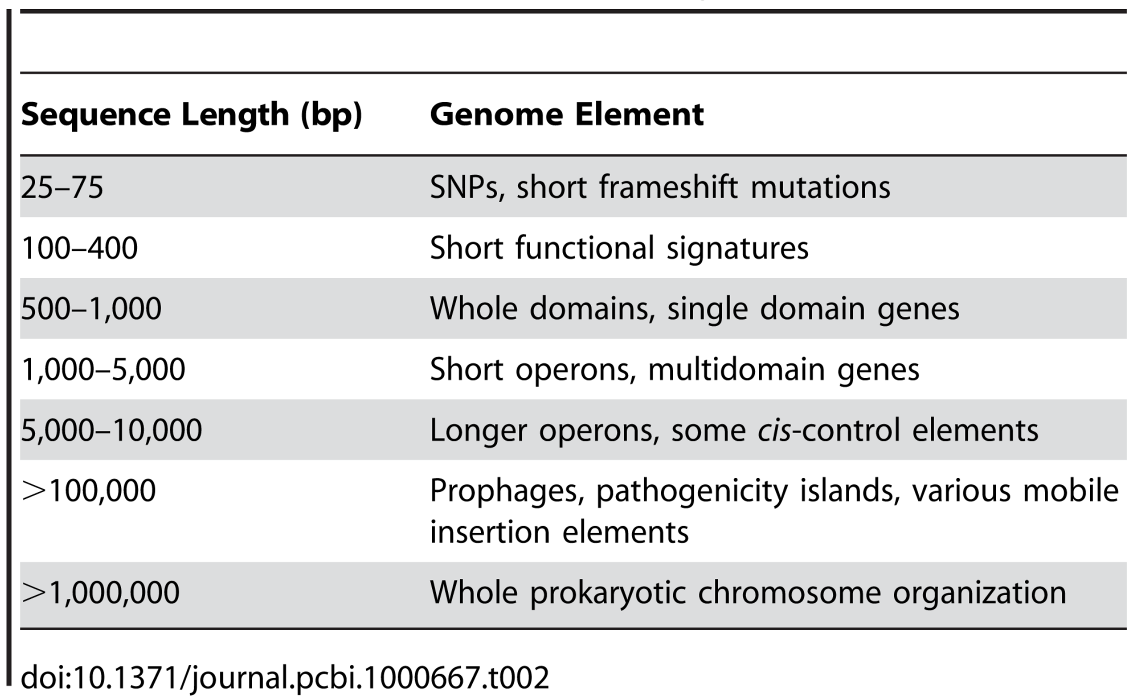
So far we've covered the steps of a metagenomic experiment to the point of obtaining sequence data. What is done with these millions (or even billions) of reads? It depends on the goals of the study, since different sequence lengths generally provide different types of information.

There is a simple correlation between the length of sequence fragments obtained and the complexity of the functional genomic details that can be gleaned from it.

By assembling these shorter sequences into larger contigs we can attempt to predict complete genes or even close whole chromosomes (in simple populations.)

Both assembly and gene prediction processes are complicated by the nature of metagenomic data.

The information contained in different lengths of DNA.



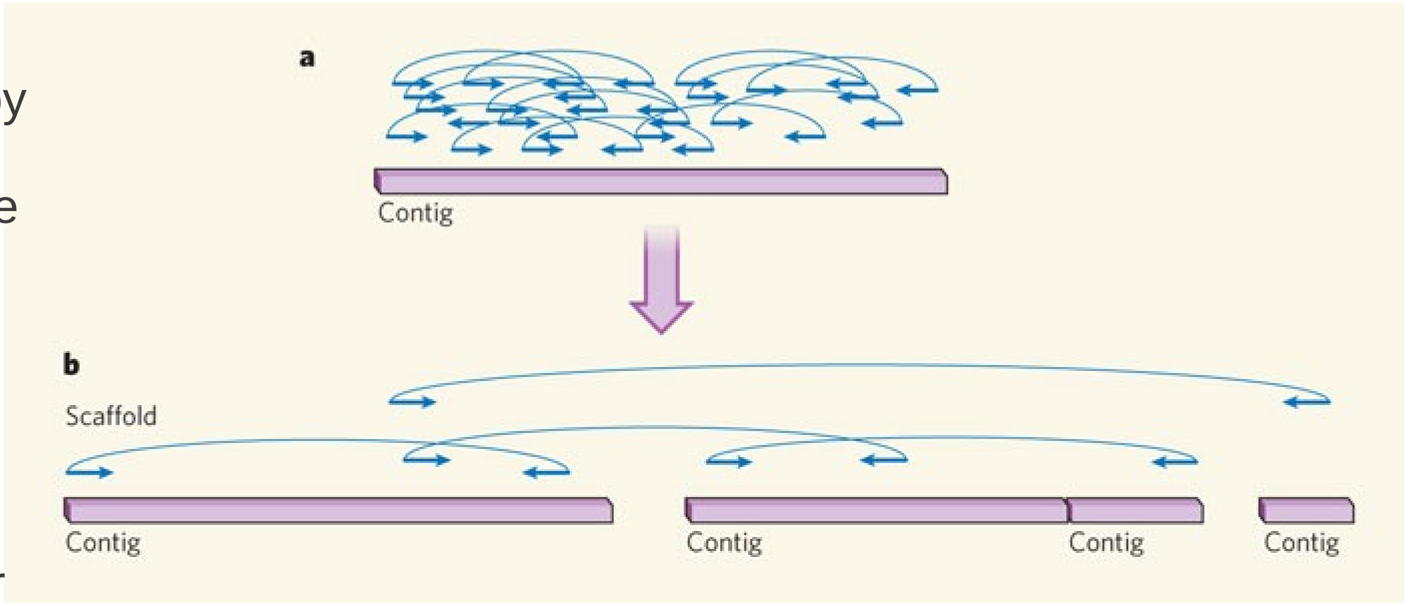
Sequence Length (bp)	Genome Element
25–75	SNPs, short frameshift mutations
100–400	Short functional signatures
500–1,000	Whole domains, single domain genes
1,000–5,000	Short operons, multidomain genes
5,000–10,000	Longer operons, some *cis*-control elements
>100,000	Prophages, pathogenicity islands, various mobile insertion elements
>1,000,000	Whole prokaryotic chromosome organization

doi:10.1371/journal.pcbi.1000667.t002

Traditional assembly

In simple terms, single-genome assembly is done by finding matching overlaps between reads in an iterative fashion.

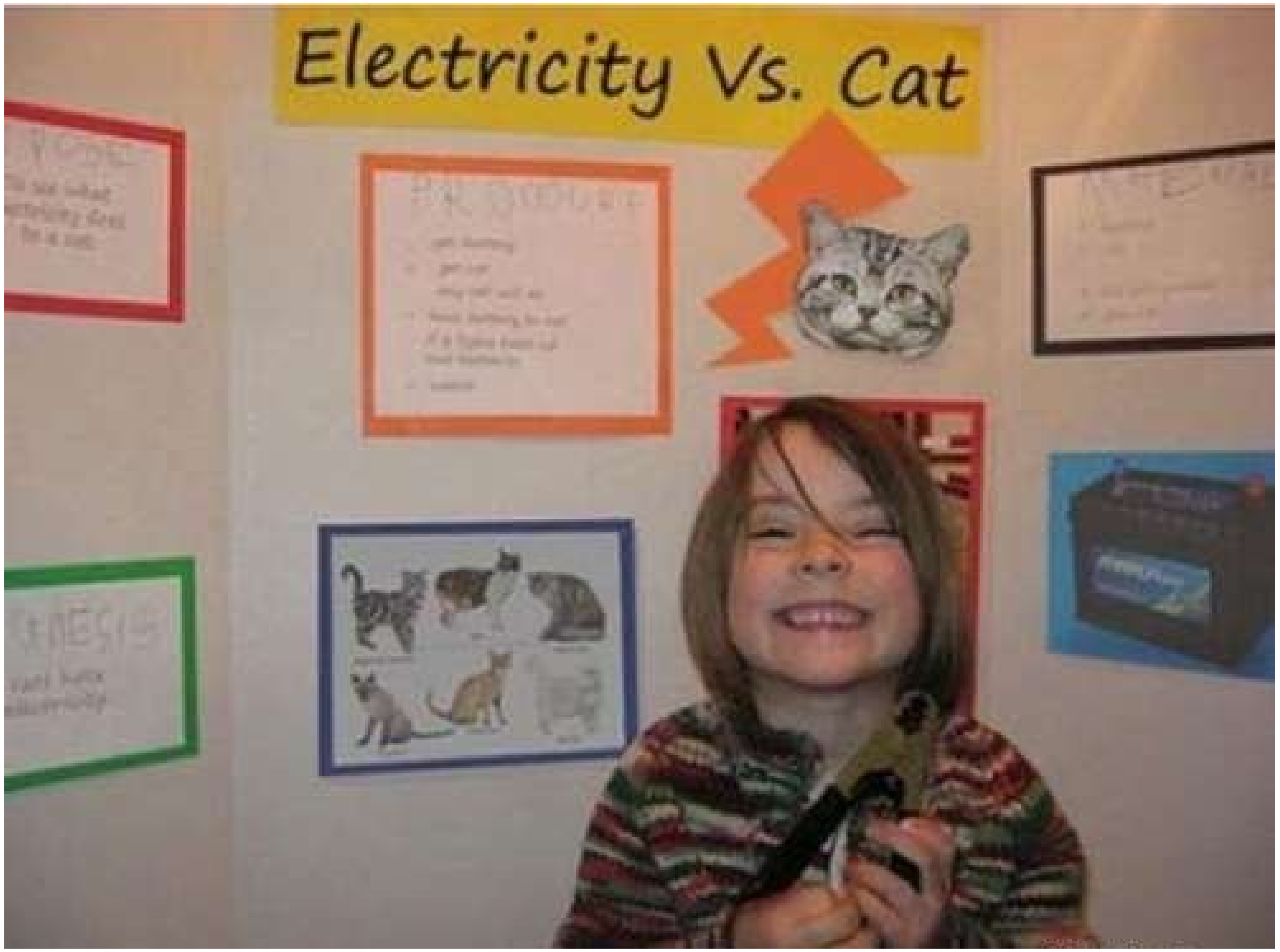
Most modern sequencers provide “paired-end” reads with at least an estimated distance between them. If this overlaps, you get longer effective read length.



Each 'contig' joined by individual overlapping reads can then be assembled further into 'scaffolds' when two different mates of a read pairs are pointing off the ends of two different contigs, as the illustration above shows.

This sort of assembly has been done for decades It is complicated by metagenomic data sets in both scale of reads to assemble and the composition of the samples.

This week's theme: Science is fun.



Metagenomic assembly challenges

The first generation of sequence assemblers were designed to assemble single, clonal samples, not mixed populations.

Metagenomics assemblers generally take one of two approaches:

Reference-based assembly

The easier of the two, this works well when reference genomes available closely match those in the sample.

Differences of the sample to the reference, such as large insertions, deletions or polymorphisms can cause the assembly to be fragmented or miss divergent regions.

These algorithms are typically fast and efficient, and can be run on a laptop in a few hours.

Newbler and AMOS are examples here.

De novo assembly

Most of the assembly tools here are written to handle large amounts of reads and are based on de Bruijn graphs (see next slide.)

They are much more resource-intensive, often taking 10s-100s of GB of RAM and multiple compute cores for a few days.

See this week's paper assignments for a review, but the most popular applications here are MetaVelvet (which deals with non-clonality of natural populations) and SOAPdenovo.

Thomas, et al. Micr. Inform. & Exp. 2012

What are de Bruijn graphs?

There are different forms of this sort of graph as applied to genomic assembly, but the construction of most start with transforming the input reads into a K-mer graph, where K is some length of sequence.

The calculation of overlaps into the graph is computationally expensive.

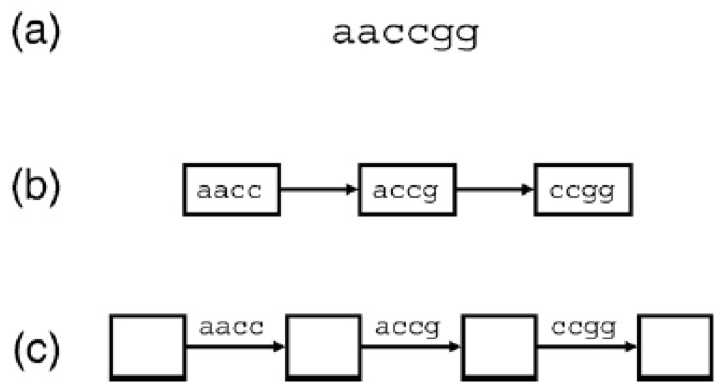


Fig. 1. A read represented by K-mer graphs. (a) The read is represented by two types of K-mer graph with K = 4. Larger values of K are used for real data. (b) The graph has a node for every K-mer in the read plus a directed edge for every pair of K-mers that overlap by K-1 bases in the read. (c) An equivalent graph has an edge for every K-mer in the read and the nodes implicitly represent overlaps of K-1 bases. In these examples, the paths are simple because the value K = 4 is larger than the 2 bp repeats in the read. The read sequence is easily reconstructed from the path in either graph.

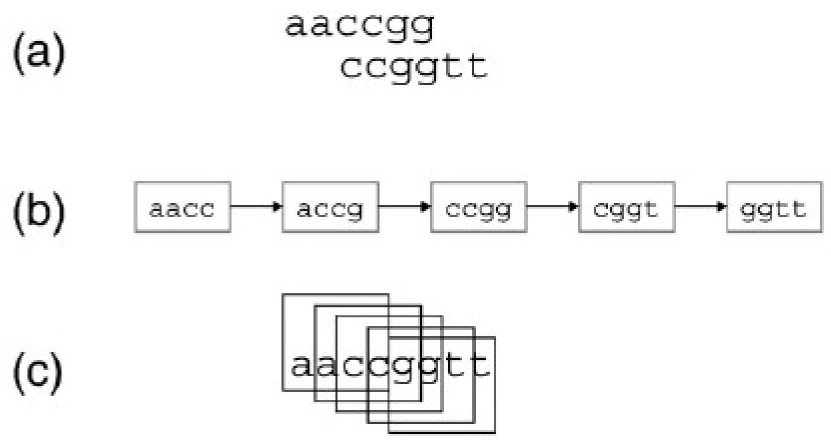


Fig. 2. A pair-wise overlap represented by a K-mer graph. (a) Two reads have an error-free overlap of 4 bases. (b) One K-mer graph, with K = 4, represents both reads. The pair-wise alignment is a by-product of the graph construction. (c) The simple path through the graph implies a contig whose consensus sequence is easily reconstructed from the path.

Most de Bruijn-type assemblers try to identify, within the entire graph, a subgraph that represents related genomes. Those subgraphs or subsets are then resolved to build a consensus sequence of the genomes.

The devil is in the details here, and assembly is really just a graph reduction problem.

Thomas, et al. Micr. Inform. & Exp. 2012

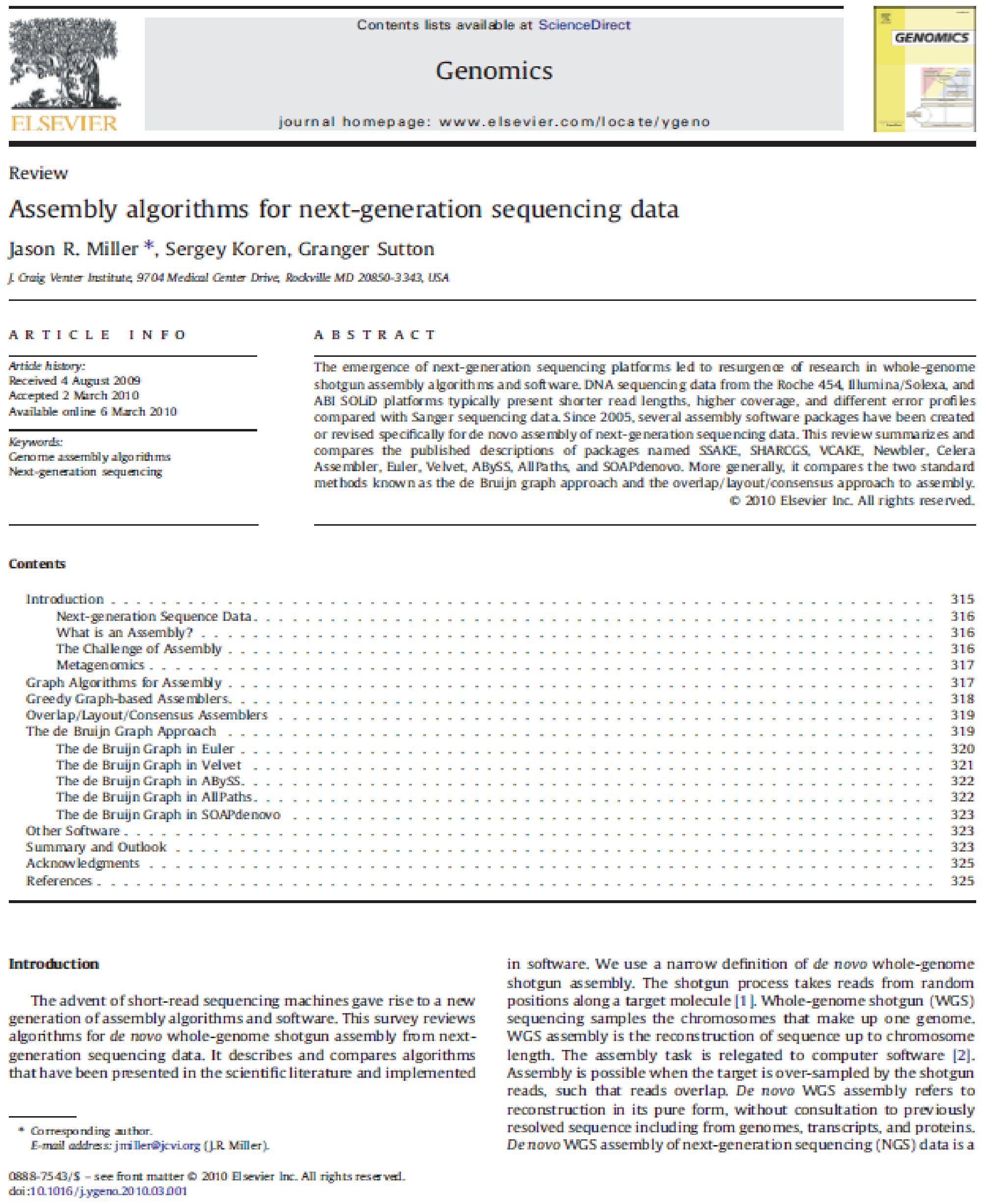
Genomics. 2010 June; 95(6): 315–327.

More on de Bruijn graphs

I could easily do a full week just on these data structures. If you want to dig into the CS bits of how these work, see the excellent paper on the right.

They are explained generally and then the variations and reduction employed by several different assemblers is compared.

I would recommend everyone at least read the section entitled "Graph Algorithms for Assembly", and then go deeper depending on your interest in the internals of assembly.



Contents lists available at ScienceDirect
Genomics
journal homepage: www.elsevier.com/locate/ygeno

Review

Assembly algorithms for next-generation sequencing data

Jason R. Miller *, Sergey Koren, Granger Sutton
J. Craig Venter Institute, 9704 Medical Center Drive, Rockville MD 20850-3343, USA

A R T I C L E   I N F O

Article history:
Received 4 August 2009
Accepted 2 March 2010
Available online 6 March 2010

Keywords:
Genome assembly algorithms
Next-generation sequencing

A B S T R A C T

The emergence of next-generation sequencing platforms led to resurgence of research in whole-genome shotgun assembly algorithms and software. DNA sequencing data from the Roche 454, Illumina/Solexa, and ABI SOLiD platforms typically present shorter read lengths, higher coverage, and different error profiles compared with Sanger sequencing data. Since 2005, several assembly software packages have been created or revised specifically for de novo assembly of next-generation sequencing data. This review summarizes and compares the published descriptions of packages named SSAKE, SHARCGS, VCAKE, Newbler, Celera Assembler, Euler, Velvet, ABySS, AllPaths, and SOAPdenovo. More generally, it compares the two standard methods known as the de Bruijn graph approach and the overlap/layout/consensus approach to assembly.
© 2010 Elsevier Inc. All rights reserved.

Contents

Introduction

The advent of short-read sequencing machines gave rise to a new generation of assembly algorithms and software. This survey reviews algorithms for *de novo* whole-genome shotgun assembly from next-generation sequencing data. It describes and compares algorithms that have been presented in the scientific literature and implemented in software. We use a narrow definition of *de novo* whole-genome shotgun assembly. The shotgun process takes reads from random positions along a target molecule [1]. Whole-genome shotgun (WGS) sequencing samples the chromosomes that make up one genome. WGS assembly is the reconstruction of sequence up to chromosome length. The assembly task is relegated to computer software [2]. Assembly is possible when the target is over-sampled by the shotgun reads, such that reads overlap. *De novo* WGS assembly refers to reconstruction in its pure form, without consultation to previously resolved sequence including from genomes, transcripts, and proteins. *De novo* WGS assembly of next-generation sequencing (NGS) data is a

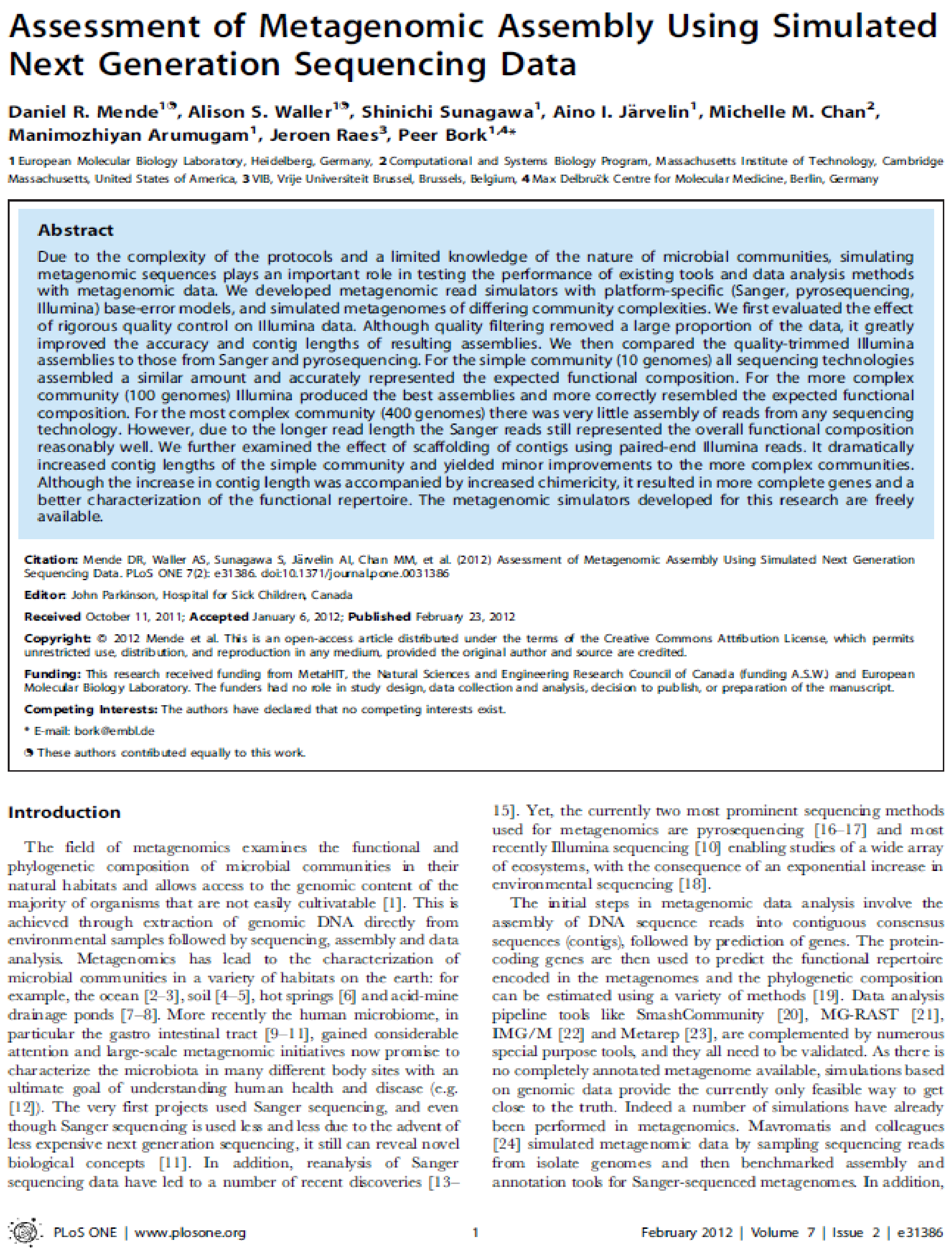
* Corresponding author.
E-mail address: jmiller@jcvi.org (J.R. Miller).

0888-7543/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ygeno.2010.03.001

Genomics. 2010 June; 95(6): 315–327.

Graph theory, for some.

The importance of controls



Assessment of Metagenomic Assembly Using Simulated Next Generation Sequencing Data

Daniel R. Mende[1]◔, Alison S. Waller[1]◔, Shinichi Sunagawa[1], Aino I. Järvelin[1], Michelle M. Chan[2], Manimozhiyan Arumugam[1], Jeroen Raes[3], Peer Bork[1,4]*

1 European Molecular Biology Laboratory, Heidelberg, Germany, 2 Computational and Systems Biology Program, Massachusetts Institute of Technology, Cambridge Massachusetts, United States of America, 3 VIB, Vrije Universiteit Brussel, Brussels, Belgium, 4 Max Delbrück Centre for Molecular Medicine, Berlin, Germany

Abstract

Due to the complexity of the protocols and a limited knowledge of the nature of microbial communities, simulating metagenomic sequences plays an important role in testing the performance of existing tools and data analysis methods with metagenomic data. We developed metagenomic read simulators with platform-specific (Sanger, pyrosequencing, Illumina) base-error models, and simulated metagenomes of differing community complexities. We first evaluated the effect of rigorous quality control on Illumina data. Although quality filtering removed a large proportion of the data, it greatly improved the accuracy and contig lengths of resulting assemblies. We then compared the quality-trimmed Illumina assemblies to those from Sanger and pyrosequencing. For the simple community (10 genomes) all sequencing technologies assembled a similar amount and accurately represented the expected functional composition. For the more complex community (100 genomes) Illumina produced the best assemblies and more correctly resembled the expected functional composition. For the most complex community (400 genomes) there was very little assembly of reads from any sequencing technology. However, due to the longer read length the Sanger reads still represented the overall functional composition reasonably well. We further examined the effect of scaffolding of contigs using paired-end Illumina reads. It dramatically increased contig lengths of the simple community and yielded minor improvements to the more complex communities. Although the increase in contig length was accompanied by increased chimericity, it resulted in more complete genes and a better characterization of the functional repertoire. The metagenomic simulators developed for this research are freely available.

Citation: Mende DR, Waller AS, Sunagawa S, Järvelin AI, Chan MM, et al. (2012) Assessment of Metagenomic Assembly Using Simulated Next Generation Sequencing Data. PLoS ONE 7(2): e31386. doi:10.1371/journal.pone.0031386

Editor: John Parkinson, Hospital for Sick Children, Canada

Received October 11, 2011; **Accepted** January 6, 2012; **Published** February 23, 2012

Funding: This research received funding from MetaHIT, the Natural Sciences and Engineering Research Council of Canada (funding A.S.W.) and European Molecular Biology Laboratory. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: bork@embl.de

◔ These authors contributed equally to this work.

Introduction

The field of metagenomics examines the functional and phylogenetic composition of microbial communities in their natural habitats and allows access to the genomic content of the majority of organisms that are not easily cultivatable [1]. This is achieved through extraction of genomic DNA directly from environmental samples followed by sequencing, assembly and data analysis. Metagenomics has lead to the characterization of microbial communities in a variety of habitats on the earth: for example, the ocean [2–3], soil [4–5], hot springs [6] and acid-mine drainage ponds [7–8]. More recently the human microbiome, in particular the gastro intestinal tract [9–11], gained considerable attention and large-scale metagenomic initiatives now promise to characterize the microbiota in many different body sites with an ultimate goal of understanding human health and disease (e.g. [12]). The very first projects used Sanger sequencing, and even though Sanger sequencing is used less and less due to the advent of less expensive next generation sequencing, it still can reveal novel biological concepts [11]. In addition, reanalysis of Sanger sequencing data have led to a number of recent discoveries [13–15]. Yet, the currently two most prominent sequencing methods used for metagenomics are pyrosequencing [16–17] and most recently Illumina sequencing [10] enabling studies of a wide array of ecosystems, with the consequence of an exponential increase in environmental sequencing [18].

The initial steps in metagenomic data analysis involve the assembly of DNA sequence reads into contiguous consensus sequences (contigs), followed by prediction of genes. The protein-coding genes are then used to predict the functional repertoire encoded in the metagenomes and the phylogenetic composition can be estimated using a variety of methods [19]. Data analysis pipeline tools like SmashCommunity [20], MG-RAST [21], IMG/M [22] and Metarep [23], are complemented by numerous special purpose tools, and they all need to be validated. As there is no completely annotated metagenome available, simulations based on genomic data provide the currently only feasible way to get close to the truth. Indeed a number of simulations have already been performed in metagenomics. Mavromatis and colleagues [24] simulated metagenomic data by sampling sequencing reads from isolate genomes and then benchmarked assembly and annotation tools for Sanger-sequenced metagenomes. In addition,

It's difficult to formally evaluate anything well without a controlled standard on which to form comparisons.

Before next-gen sequencing, the FAMeS Data Sets served here, providing defined artificial metagenomic sequence pools with three different levels of taxonomic complexity (PMID: 17468765).

This study by Mende, et al. (one of your assigned readings this week) improves upon this idea with NGS data sources. They developed read simulators from three different sequencing platforms which included base-error modeling for each.

Then they used these to generate simulated metagenomes of different community complexities.

Mende, et al.   (cont'd)

Once created, the simulated metagenomes were used to benchmark currently used assembly protocols with data from Sanger, pyrosequencing and Illumina reads.

Because there was no standard treatment of Illumina reads with regard to quality filtering, and due to the rise of Illumina sequencing for metagenomics, they also evaluated the impact pre-screening the reads based on quality data.

Chimerism is the formation of an invalid contig based on the overlap of two or more reads from different genomes. It is usually a concern when doing assembly, and the authors tested and scored it here.

Looking forward, the authors also tested how well the product of each protocol descibed the functional capability of the sample. This was done first with gene prediction using MetaGeneMark and then generating COG assignments using BLAST. This is a quick, though not particularly robust, way of creating a functional profile of the genes in your set.

Finally, they also tested the affect of scaffolding on assemblies of different communities.

See the forums in Blackboard for a discussion of the results here.

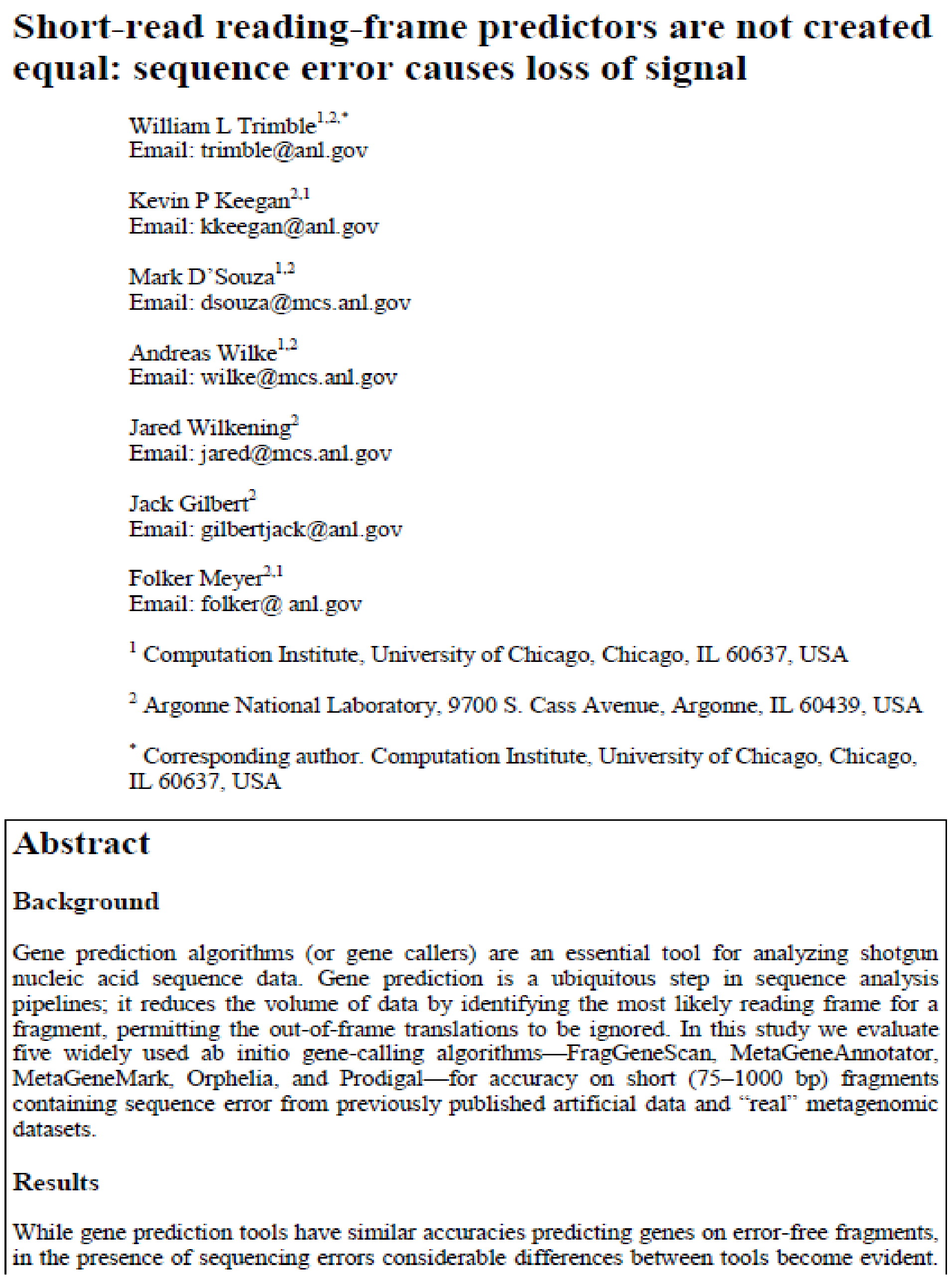
Gene prediction

After sequence assembly the next step for many analyses is gene prediction. Your second assigned paper for this week's lesson is this comparison of de novo gene prediction tools.

Specifically, those compared are:

- FragGeneScan
- MetaGeneAnnotator
- MetaGeneMark
- Orphelia
- Prodigal

Finding: FragGeneScan finds more prokaryotic coding regions than the others, but this improved detection of genes in error-containing fragments comes at the cost of much lower (50%) specificity and overprediction of genes in noncoding regions. What do you think of this?



Short-read reading-frame predictors are not created equal: sequence error causes loss of signal

William L Trimble[1,2,*]
Email: trimble@anl.gov

Kevin P Keegan[2,1]
Email: kkeegan@anl.gov

Mark D`Souza[1,2]
Email: dsouza@mcs.anl.gov

Andreas Wilke[1,2]
Email: wilke@mcs.anl.gov

Jared Wilkening[2]
Email: jared@mcs.anl.gov

Jack Gilbert[2]
Email: gilbertjack@anl.gov

Folker Meyer[2,1]
Email: folker@ anl.gov

[1] Computation Institute, University of Chicago, Chicago, IL 60637, USA

[2] Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA

[*] Corresponding author. Computation Institute, University of Chicago, Chicago, IL 60637, USA

Abstract

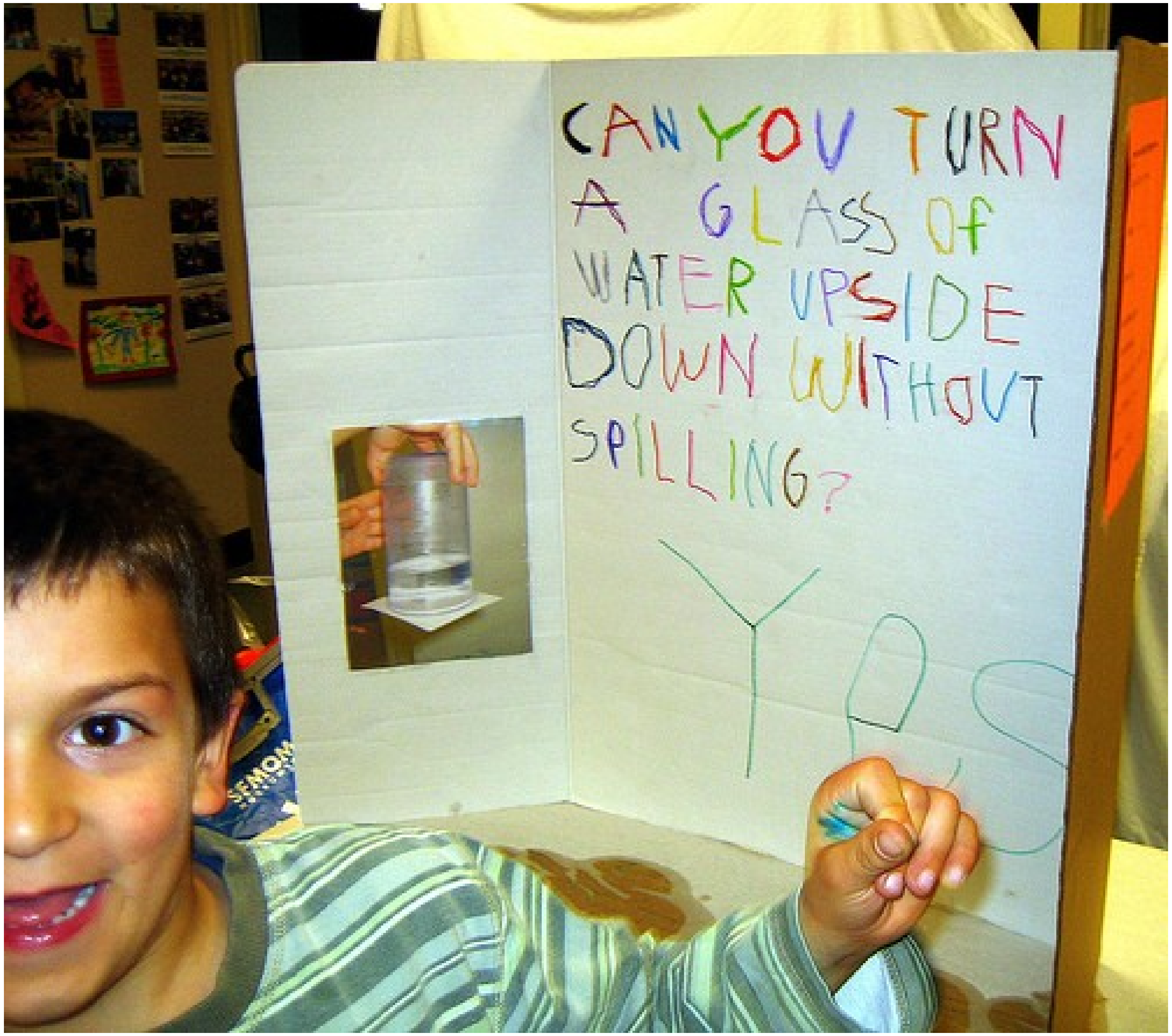
Background

Gene prediction algorithms (or gene callers) are an essential tool for analyzing shotgun nucleic acid sequence data. Gene prediction is a ubiquitous step in sequence analysis pipelines; it reduces the volume of data by identifying the most likely reading frame for a fragment, permitting the out-of-frame translations to be ignored. In this study we evaluate five widely used ab initio gene-calling algorithms—FragGeneScan, MetaGeneAnnotator, MetaGeneMark, Orphelia, and Prodigal—for accuracy on short (75–1000 bp) fragments containing sequence error from previously published artificial data and "real" metagenomic datasets.

Results

While gene prediction tools have similar accuracies predicting genes on error-free fragments, in the presence of sequencing errors considerable differences between tools become evident.

Hypothesis? Methods?



Grid computing and DIAG

We are rapidly shifting into the realm of needing more significant compute resources than we have been using for our small scripting tasks. Gene prediction can be a memory-intensive process and, to a greater extent, so is genome assembly.

You will be able to learn to run these steps yourself using DIAG, which provides:

High-throughput nodes: 1500 physical cores with 48GB RAM and 3.5TB local storage per node.

High-performance nodes: 5 nodes, each with 160 physical cores, 1TB local storage and up to 1TB RAM.

Job scheduling system:

Oracle Grid Engine

To manage this, you'll need a mechanism to submit, monitor and check your jobs to these nodes. There are several mechanisms for this, but we'll start with Oracle's Grid Engine, which you can use to submit jobs directly from the command-line.

Using SGE, submitting jobs

Before you can use SGE you need to set up your environment for it. After you've logged in, you can type this at your terminal:

```
    source /diag/software/sge-root/diag/common/settings.sh
```

Or, if you add that line to the end of your ~/.bashrc file it will happen automatically for you each time you log in.

With that done, here are example commands:

Submit a script file (which can't have arguments):

```
    qsub -P diag /path/to/yourscript.sh
```

Submit an arbitrary command (can have arguments):

```
    qsub -P diag -b y /path/to/yourscript.pl --input=foo --format=fasta
```

The “-P” option specifies a project code, which you should leave as 'diag' for now.
The -b y option treats what follows as a binary to execute with options rather than looking *into* the passed script for commands. In my experience, you'll probably almost always submit using the second form.

Using SGE, advanced submission

The previous two examples are fine for quick submissions, but for memory-intensive jobs with long run times we'll a need several more. The bold parts are an example of options you'd actually type, followed by a description. All of these except the -P are optional.

-P diag	sets the SGE project code
-q highmem.q	Restricts a job to a named 'queue' of nodes.
-l mem_free=24G	Specify the max memory needed for your job.
-o /path/job.stdout	The STDOUT of your job.
-e /path/job.stderr	The STDERR of your job. These are both important to explicitly save else it might be difficult to debug your job failures.
-v MYLIB,PATH	Any environmental variables you've set that you want available to your job.

<u>General syntax</u>:

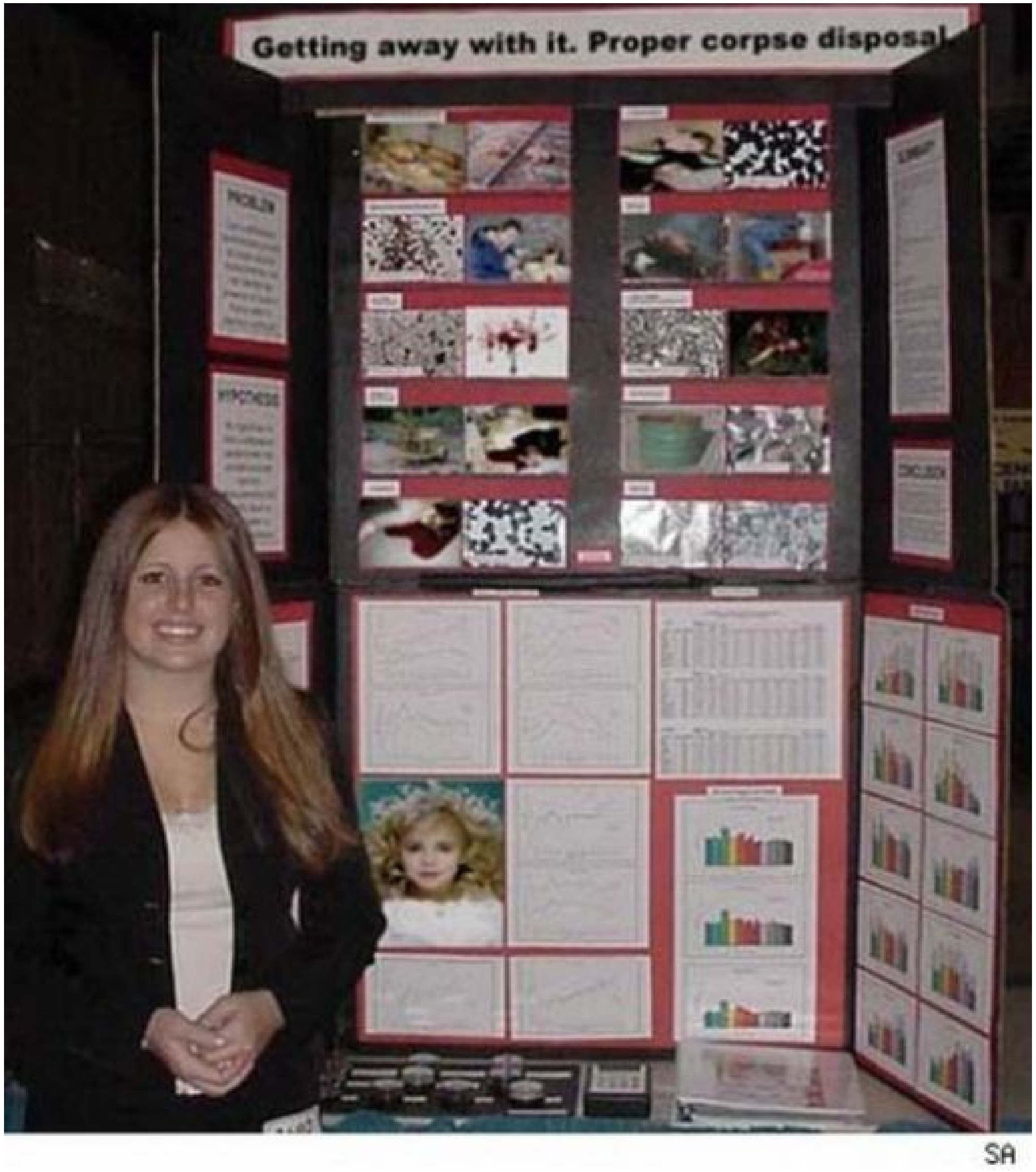
qsub <*sge options*> -b y <*your command's options*>

Detailed example:

```
qsub -P diag -l mem_free=10G -o
/home/jorvis/job.stdout -e
/home/jorvis/job.stderr -v PATH -b y
/home/jorvis/svn/biocode/convert_fastq_
to_fasta.py -i /tmp/foo.fq -o
/tmp/foo.fna
```

Notice that there are two -o options in the example above. One is within the SGE options, and the other in my scripts options. Anything after the “-b y” is assumed to be a script option and ignored by SGE.

Well that's horrible.



Using SGE, after submission

When you use the qsub command to submit a job, it will give you a job ID right away. Given an example job ID 4415, you can:

Get details about a job that's pending or running:

```
qstat -j 4415
```

Get details about a job no longer in the queue:

```
qacct -j 4415
```

See a list of all jobs by user name:

```
qstat -u jorvis
```

See everyone's jobs:

```
qstat -u '*'
```

Kill a job by its ID:

```
qdel 4415
```

You'll probably use all these throughout the rest of the semester. Feel free to ask questions as we go so your work doesn't get delayed.

Further reading

An Eulerian path approach to DNA fragment assembly

Proc Natl Acad Sci U S A. 2001 August 14; 98(17): 9748–9753.
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC55524/

Assembly algorithms for next-generation sequencing data

Genomics 95 (2010) 315-327.
http://www.sciencedirect.com/science/article/pii/S0888754310000492#

Ab initio gene identification in metagenomic sequences

Nucl. Acids Res. (2010) 38 (12): e132.
http://nar.oxfordjournals.org/content/38/12/e132.full