

Assembly of the *Ariolimax dolicophallus* genome with Discover *de novo*

**Chris Eisenhart, Robert Calef, Natasha Dudek,
Gepoliano Chaves**

Overview

- Introduction
- Pair correction and filling
- Assembly theory overview
- User experience and installation
- Sequence preprocessing
- Documentation and results

Discover *de novo*

- Discover *de novo* is a large/small genome assembler
- Developed by the Broad Institute (MA)
 - <http://www.broadinstitute.org/software/discover/blog/>
- Generates accurate and complete assemblies

Discover *de novo*

- Discover *de novo* requires a single Illumina fragment library (paired end)
- A PCR-free protocol, ~450bp insert size, and ~60X coverage are recommended (SPRI beads).
- 250bp (recommended for the assembler) or higher paired reads will be created by the sequencing machine

Discover *de novo*

- Reads as short as 150bp may work with Discover *de novo*, depending on fragment size and other factors.
- This will however require algorithmic modifications

Discover *de novo*

- Other sequencing technologies such as 454, SOLiD and PacBio may not be used. The assembler is restricted to Illumina.

Unipaths and unipath graphs

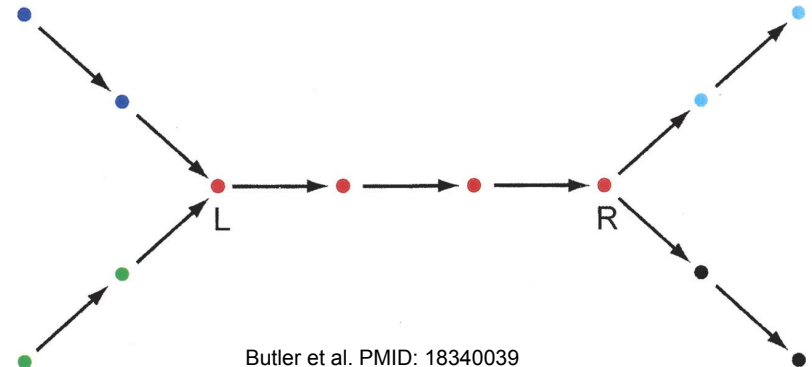
-Unipath

- An unambiguous path. Formally: Let $x_1 \dots x_n$ be a sequence of adjacent k-mers in the de Bruijn graph such that $x_1 \dots x_{n-1}$ have outdegree 1 and $x_2 \dots x_n$ have indegree one, any extension of violates this constraint

- Similar to U-U contigs in meraculous

-Unipath graph

- Like a de Bruijn graph, with unipaths for edges. Nodes are still k-mers



Stages

Stage 1: Preliminary graph

- Reads are error corrected
- Pairs are closed
- Pair closures are merged into a graph

Stage 2: Optimization of graph

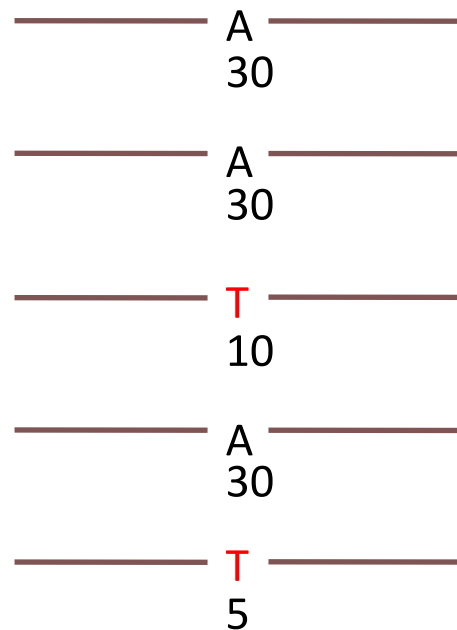
- Simplify and improve the graph

Pair correction and filling

- Precorrection 1
- Pair filling 1
- Quality score lowering
- Precorrection 2
- Pair filling 2
- Pair correction and filling

Pre correction 1

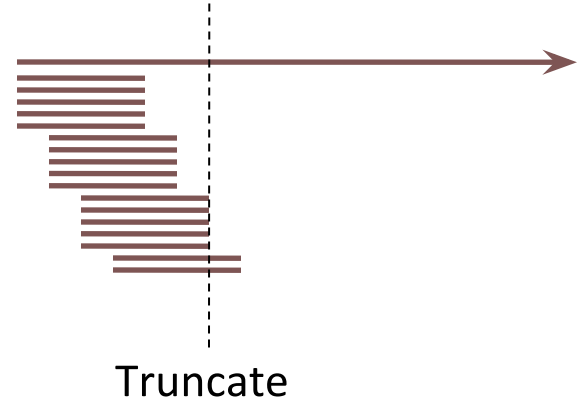
- Perform a sort to bring together 25-mers with same initial and final 12-mers
- Focus only on central base: what should the true call be?
 - Calculate sum of quality scores for each possible calls
 - Winner call = base with highest sum of quality scores
 - Loser call = base with no more than 1 call of quality 20+ and sum of quality scores must be $1/4^{\text{th}}$ that of the winner
- Repeat for all 25-mers
- Correct isolated loser calls, set quality score to 0



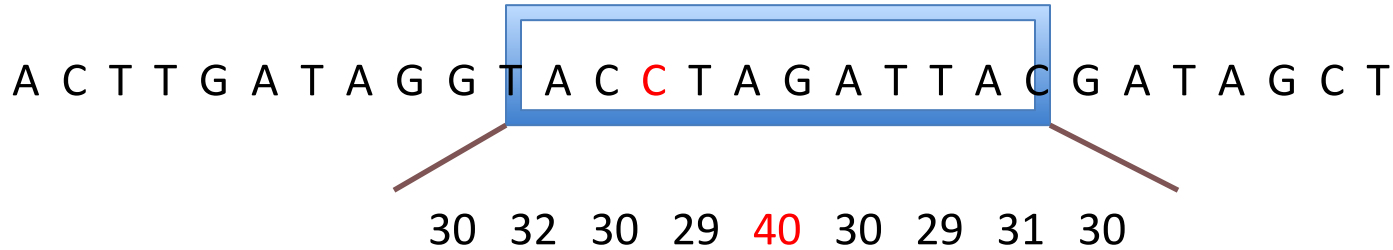
Pair filling 1

Purpose: fill pairs that are easy to define as having a unique, unambiguous closure

- Consider all 60-mers in a read that occur $\geq 5X$ in the dataset and exclude all 60-mers that occur after this within a read not meeting this criteria
- Form unipath graph according to these truncated reads

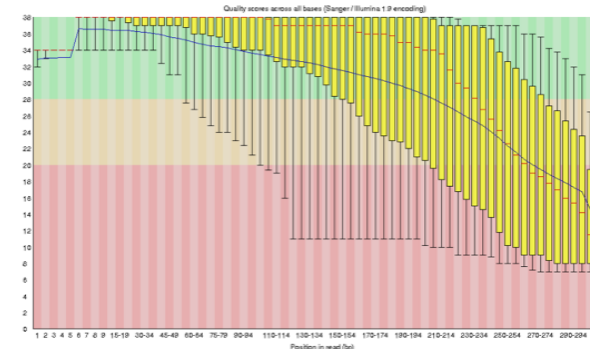


Quality score lowering



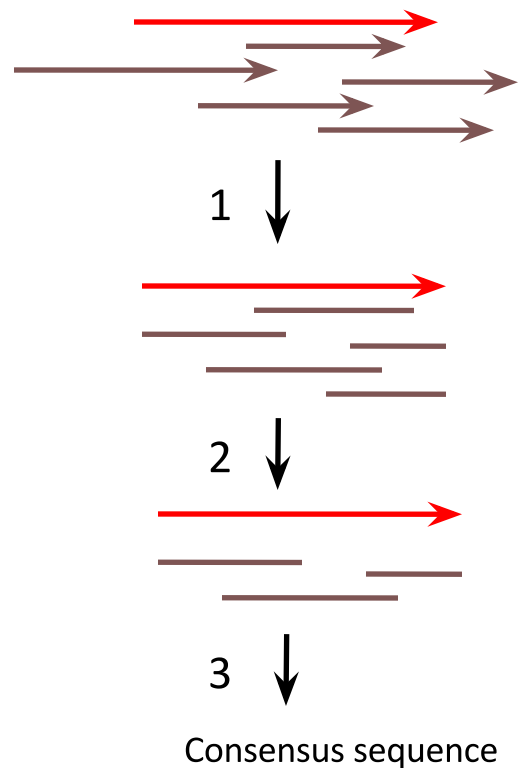
Reset to 29

Per base sequence quality
of raw reads →



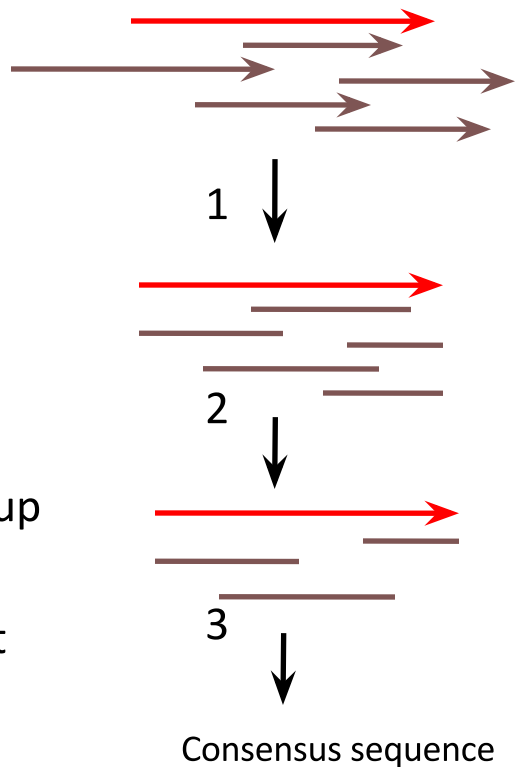
Precorrection 2

- Done twice, first with $k=24$, then $k=40$
- Take every read, one at a time = founder read
- Create stack of friends for founder
 - Gap free alignment with perfect k-mer matches
- Truncate friends extending past founder (1)
- Clean the stack (2) by removing friends with Q30 difference or high quality window difference



Pre correction 2

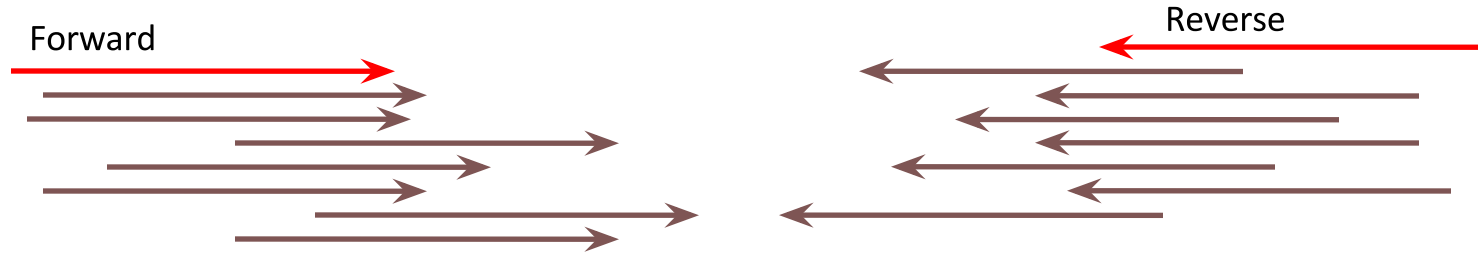
- Determine consensus sequence (3)
 - Quality scores of 1 or 2 are changed to 0.2
 - For each possible call in a column, calculate quality score sum (winner = highest sum)
 - Deduct top quality score from non-winners' quality score sum
 - Test if winner call should be accepted:
 - Winner's quality score sum must be at least 50
 - The winning sum must be at least 10X that of the runner up
 - The runner up sum must be at most 100
 - If winner is accepted and it disagrees with the founder, correct founder and set quality score to 0



Pair filling 2

- Same general method as pair filling 1
- $K=80$
- Truncation according to first spot where an 80-mer has a difference from the consensus sequence

Pair correction and filling



- Create stacks of friends for pairs of founder reads
 - Use $k=40$
 - Allow extension to the right of each read
- Do not proceed with low quality founder pairs
 - Let a = mean quality of bases in founder pair
 - Let b = mean quality of all bases in all friend reads
 - If b exceeds a by more than 20, do not proceed

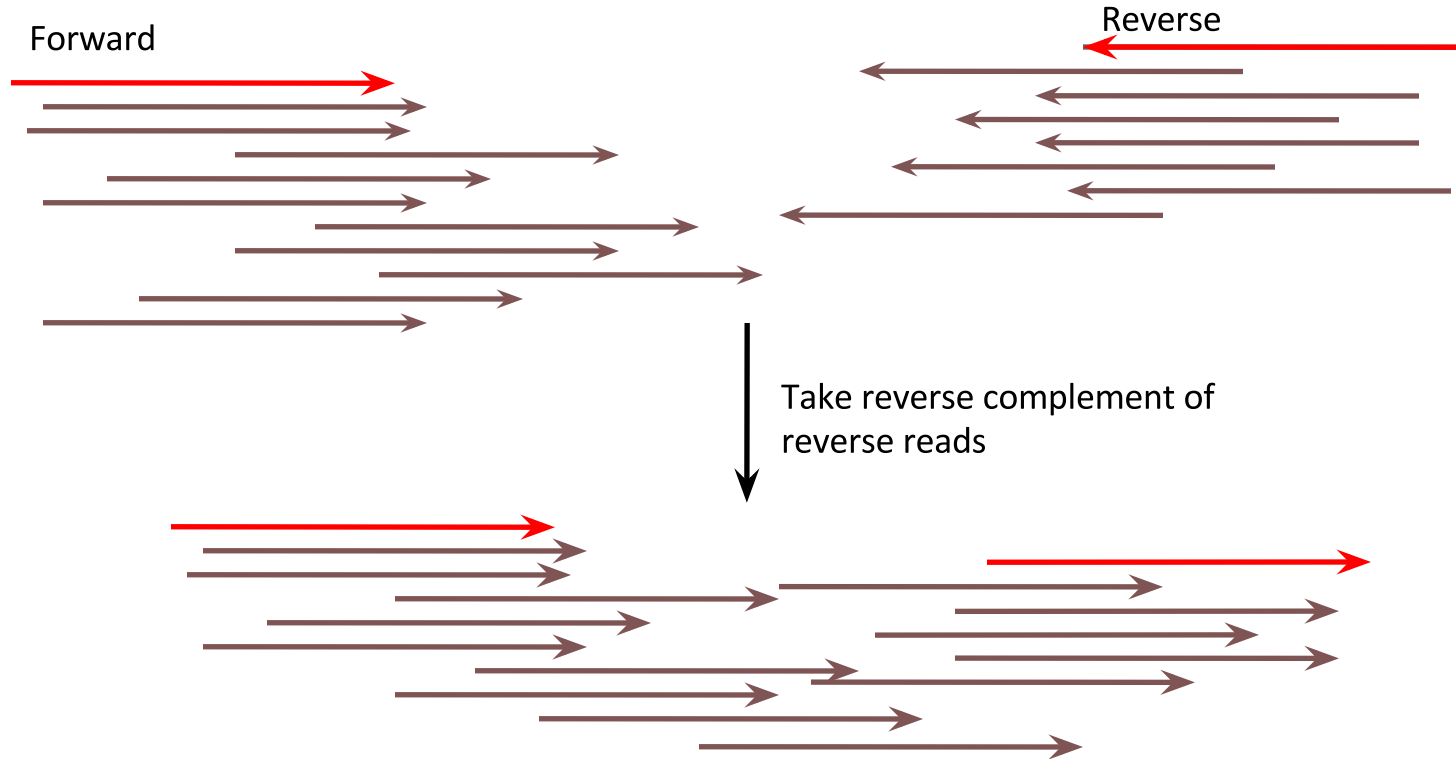
Pair correction and filling

- Remove friends with “inadequate glue”
 - Find intervals of maximal agreement with founder
 - Compress homopolymers longer than 10 to be 10
 - To be adequately glued, after compression there must be an interval of agreement of length ≥ 20
- For every base that with a quality score between 0-30, consider raising the quality score to 30 (internal to EC)
 - Examine 11-bp long windows
 1. Are there 3 friends that agree at central base and have quality ≥ 30
 2. Are there 3 friends that disagree at central base and have quality ≥ 30
 - If 1 but not 2, correct. If both 1 and 2, do not correct

Pair correction and filling

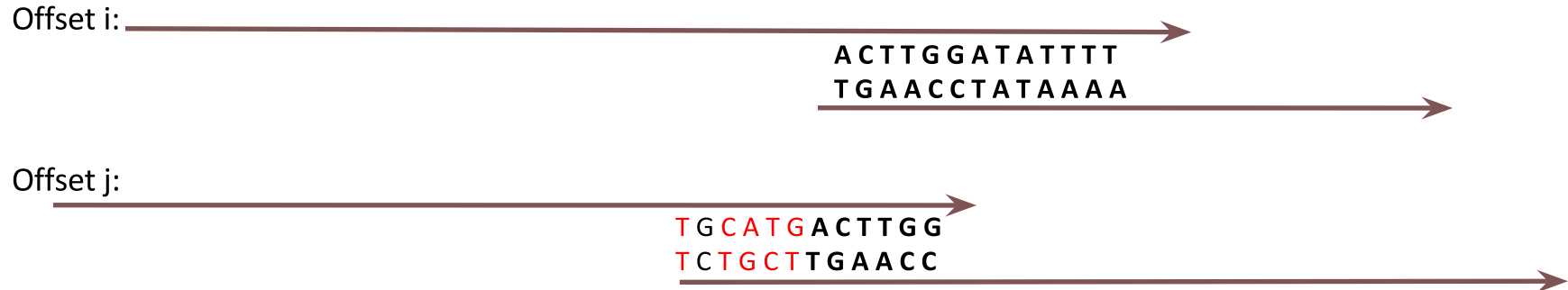
- Consider deletion of more friends based on repeating, different sequences
 - Scan 10-bp non-overlapping windows
 - If any sequence occurs as often as the founder sequence, and any single friend contains that sequence and has a quality score of over 20 at the middle bases, delete all friends with that sequence
- Clean stack by removing friends with Q30 difference with founder

Pair correction and filling

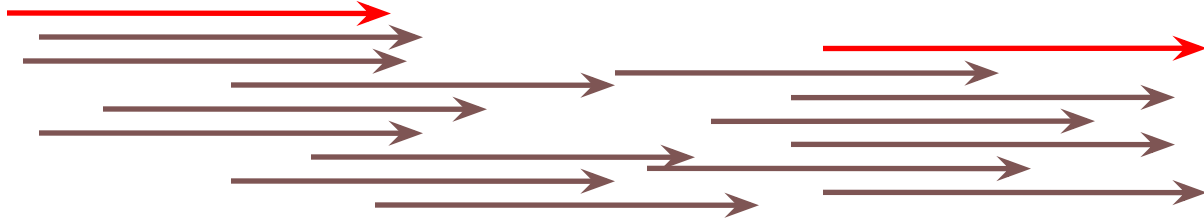


Pair correction and filling

- Compute consensus sequence for each stack (as done in Precorrection 2) and correct reads
- Compute candidate offsets by looking for overlapping 8-mers
- Candidates are evaluated based on the number of mismatches per 20bp window. High numbers of mismatches are discarded.
- Next compare offsets and allow certain offsets to invalidate others:



Pair correction and filling



- Create joint stack for each combination of potential offsets
- Raise quality scores (as previously described)
- Remove friends with Q30 difference

Pair correction and filling

- Calculate quality score sum for each call at each position
 - If the winning sum is ≥ 100 and 10X the runner up, and the runner up sum is < 100 , remove any friends that disagree and have quality ≥ 30
- Create consensus sequence (as done in Precorrection 2)
- Compute quality scores for consensus sequence
 - If a base and flanking 'd' bases agree with consensus, and are > 0 but $< 10 * \log_{10}(2d) * 0.5$, raise quality score to $10 * \log_{10}(2d) * 0.5$
 - Compute sum of quality scores for each column
 - Let x = difference between winner and runner up
 - If $x < 50$, quality score = x . If $x > 50$, quality score = 50

Pair correction and filling

Test for inconsistencies with quality scores:

- a) Look at quality score sum of runner up call.
 - If it was >100 and supported by ≥ 2 Q30 bases, declare the column inconsistent and set consensus $q=0$
- b) “Protect” certain bases in the founders.
 - Take 10 bases on the left.
 - If any founder base disagrees with the consensus and has a quality score ≥ 20 , force consensus sequence to mirror founder.
 - Repeat for 10 bases on the right.

Pair correction and filling

Test for inconsistencies with quality scores (continued):

- c) If any base on the founder has a quality score ≥ 30 , and disagrees with the consensus, set consensus $q=0$

- d) Check for inconsistencies between founders and consensus sequence
 - If the founder disagrees but the 5 flanking bases agree, and ≥ 3 other rows agree with the founder, set quality to 0

Pair correction and filling

Recover conflicted columns in consensus:
(columns with low quality score)

- Let $\text{minq_floor} = 10$ if there is more than one offset, otherwise 5
- Look at columns in the consensus that have a lower quality score than the minq_floor threshold, and for which at least 1 founder has $q \geq 2$
- For a given base call that disagrees with the founder, delete all rows containing that base call.
- Recompute consensus and consensus quality scores for the joint stack

Pair correction and filling

Decide if a closure is accepted:

1. The minimum consensus quality must be ≥ 10
2. It must be possible to walk across the consensus using only intervals of perfect overlap of length ≥ 40 , while requiring that consecutive intervals overlap by ≥ 30

Pair correction and filling

Decide if a closure is accepted:

1. The minimum consensus quality must be ≥ 10
2. It must be possible to walk across the consensus using only intervals of perfect overlap of length ≥ 40 , while requiring that consecutive intervals overlap by ≥ 30

Report closures:

1. Examine consensus sequences from all offsets, report left and rightmost maximal agreeing segment
2. If left and rightmost segments are the same, just report one

Pair correction and filling

Identify pairs that were not closed but that bridge gaps:



- Look for 40-mers that are near the edge of the gap (within 200bp)
- For all “special” read pairs with such a 40-mer, redo error correction and pair filling step with more lenient settings

Assembly theory

-Graph formation

- Initial graph is exactly the unipath graph

-Graph simplification and improvement

Simplification and improvement

- Reverse complement removal
- Hanging end removal
- Remove small components
- Delete low coverage edges
- Assembly unwinding

Simplification and improvement

- Pull apart simple branches
- Pull apart complex branches
- Bubble popping
- Graph reconstruction
- Make and unroll loops

Simplification and improvement

- Remove weakly supported loops
- Delete weakly competing edges
- Graph cleaning using the uncorrected reads
- Gulp edges
- Orient assembly to reference

User experience: Installation

- Prerequisites:

- GCC 4.7+

- jemalloc 3.6.0+

- Standard pipeline:

configure → make → make install

User experience: GCC

The wrong way:

Compile and link the dependencies:

- GMP, MPFR MPC (floating point arithmetic libraries)
- Each in a separate directory

Attempt to configure with libraries all over the place:

```
./configure --with-gmp=/some/silly/path/gmp --with-mpfr=/some/silly/path/mpfr --with-mpc=/some/silly/path/mpc
```

This is silly and causes major problems for anyone who doesn't understand how dynamic linkers find libraries at runtime. Do not do this.

User experience: GCC

The right way:

Let GCC do all the work:

```
tar xzf gcc-4.9.2.tar.gz  
cd gcc-4.9.2
```

```
./contrib/download_prerequisites
```

 <- Downloads and configures prerequisites

```
cd ..  
mkdir objdir  
cd objdir  
$PWD/../../gcc-4.9.2/configure --prefix=$HOME/gcc-4.9.2  
make  
make install
```

User experience: Installation

- Prerequisites:

- GCC 4.7+ ✓
- jemalloc 3.6.0+

- Jemalloc:

configure → make → make install

```
./configure --prefix=$HOME/jemalloc/  
make  
make install
```

User experience: Installation

- Prerequisites:

- GCC 4.7+ ✓

- jemalloc 3.6.0+ ✓

- Last but not least:

```
/configure CC=$HOME/gcc-4.9.2/bin/gcc --prefix=$HOME/discover/install_dir/ --with-jemalloc=$HOME/jemalloc/lib/  
make  
make install
```

Syntax for Discover *de novo* Assembly

- Discover READS = reads.bam OUT_DIR=my_assembly
- Takes all reads in BAM file, generates an assembly and writes output to my_assembly directory
- final assembly is located in my_assembly/a.final/

User experience: Running

- Picky command line syntax
- No white space

EX.

```
READS="frac:0.05,sample:18H:./scratch/bananaSlugBAMS/SW018_S1_L \
```

```
007_001.bam+frac:0.05,sample:19M:./scratch/bananaSlugBAMS/SW019 \
```

```
_S1_L001_001.bam+frac:0.05,sample:19H:./scratch/bananaSlugBAMS/ \
```

```
SW019_S2_L008_001.bam" OUT_DIR=./scratch/bananaSlugAssemblies
```

User experience: Running

- Picky command line syntax
- Discover denovo option 'frac'
 - Lets user define a subset of the data
- Ran with 1%, 5% and 10% of the data
 - Verify assembly plausibility

Sequence preparation

- FastQC
- Preqc
- Skewer (adapter trimming)
- fastuniq (PCR duplicate removal)
- Conversion to BAM format required

fastuniq: Rationale

- Designed for PCR-free libraries
- Reduce memory footprint
- Could bias error correction

fastuniq

de novo duplicate removal

- as opposed to mapping-based (Picard)
- implemented in C

Low RAM footprint:

35.6 GB for 16.6 gigabases of data

Merge sort →

Step 1:

Raw read pairs



Importing



Sorting



Identifying duplicates



Unique read pairs

Step 2:

Step 3:

BAM conversion rationale

-Quality representation

- Fastq uses several quality score representations

-Paired data representation

- Fastq format stores read pairs in two separate files

-BAM compression

- Binning scheme makes access easier
- Compression uses less disk space

BAM conversion rationale

-BAM files offer increased documentation potential

- Optional fields can be filled
- Optional flag for identifying unaligned sequences

Fastq to BAM

- Picard toolset
- java .jar files
- Used program fastqToSam
 - Converts fastq forward and reverse files into a single bam file

Results: Fastq vs Bam

- Bam

- 3 data files
- 47 gigs total

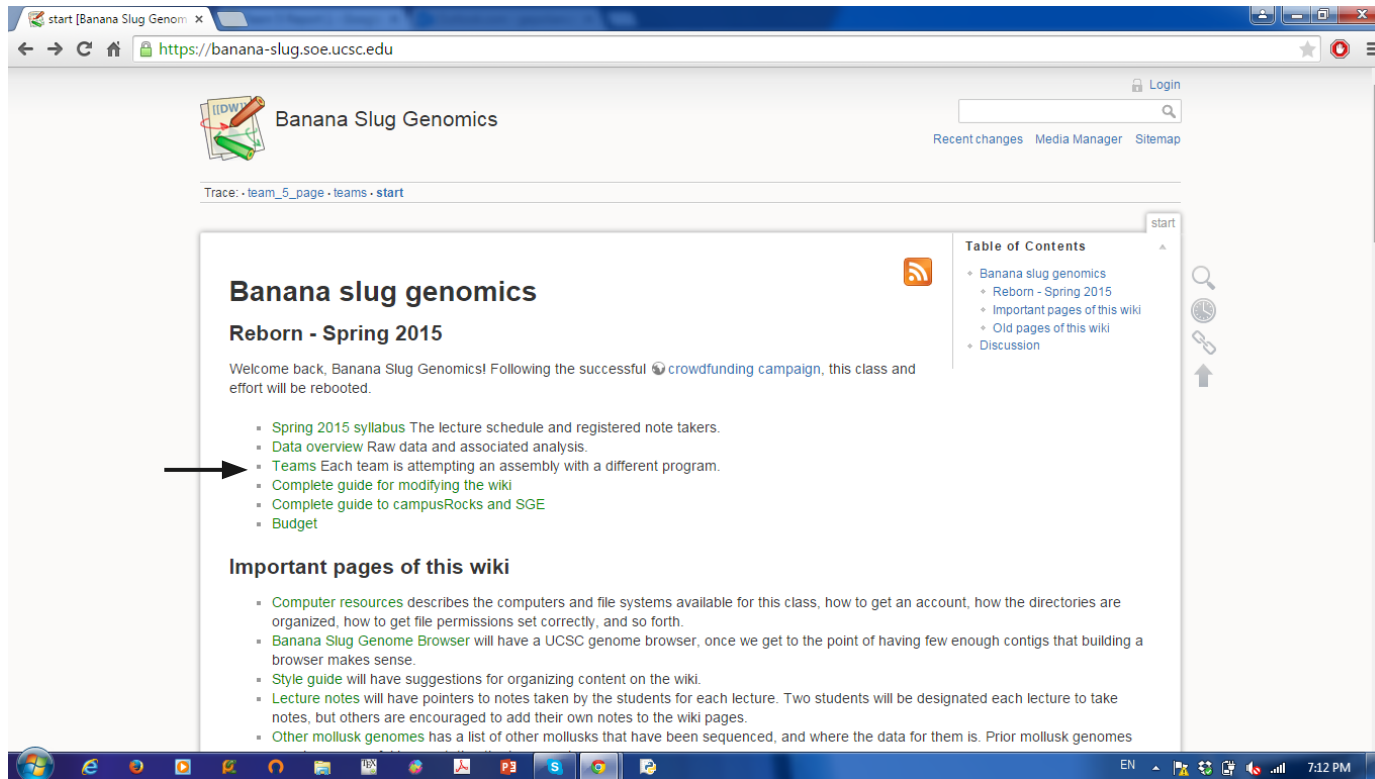
- Fastq

- 6 data files
- 55 gigs total (zipped)
- 150 gigs total (unzipped)

Documentation

- Wiki page was made
 - https://banana-slug.soe.ucsc.edu/team_5_page
- Bam files added to corresponding data pages

BME235 wiki page



The screenshot shows a web browser window displaying the Banana Slug Genomics wiki page. The browser's address bar shows the URL <https://banana-slug.soe.ucsc.edu>. The page header includes the site logo, the title "Banana Slug Genomics", a search bar, and links for "Recent changes", "Media Manager", and "Sitemap". A breadcrumb trail reads "Trace: - team_5_page - teams - start".

The main content area features the heading "Banana slug genomics" with a subheading "Reborn - Spring 2015". Below this, a welcome message states: "Welcome back, Banana Slug Genomics! Following the successful crowdfunding campaign, this class and effort will be rebooted." A list of links follows, with an arrow pointing to "Spring 2015 syllabus":

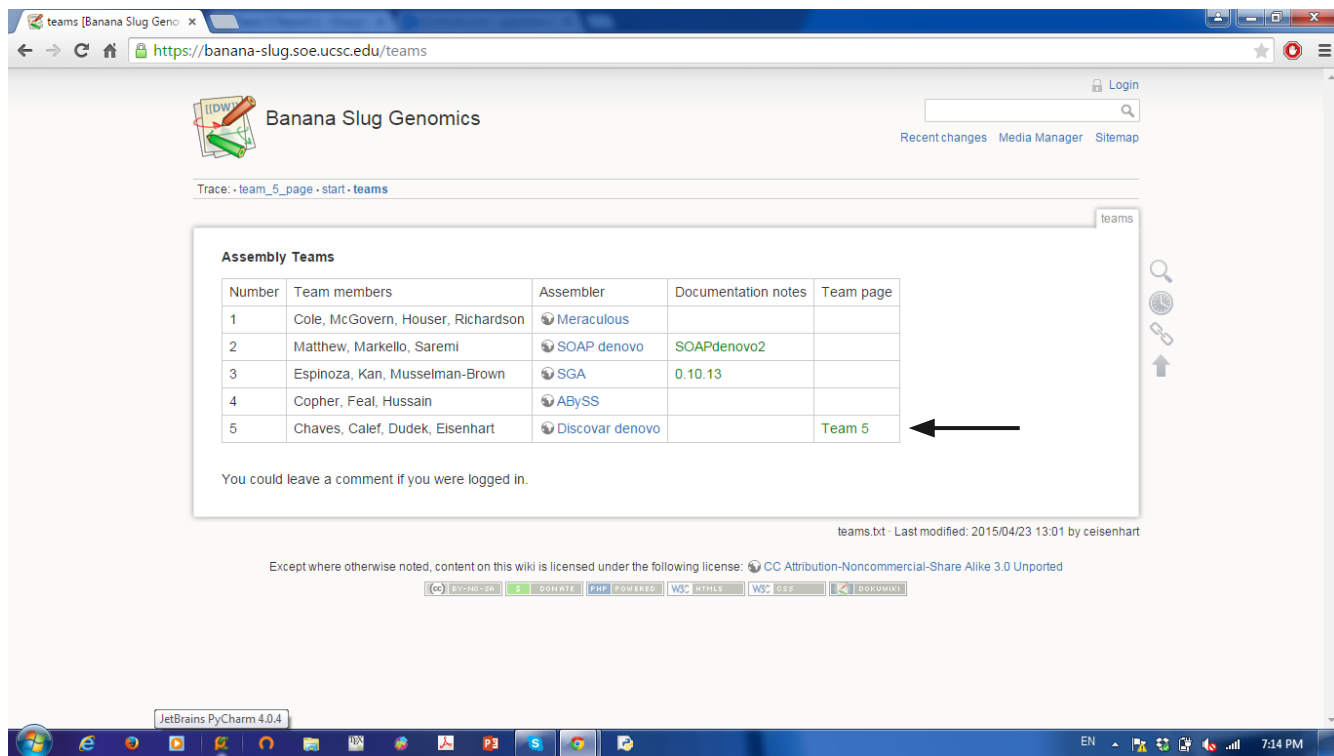
- Spring 2015 syllabus The lecture schedule and registered note takers.
- Data overview Raw data and associated analysis.
- Teams Each team is attempting an assembly with a different program.
- Complete guide for modifying the wiki
- Complete guide to campusRocks and SGE
- Budget

Below the list is the section "Important pages of this wiki" with the following links:

- Computer resources describes the computers and file systems available for this class, how to get an account, how the directories are organized, how to get file permissions set correctly, and so forth.
- Banana Slug Genome Browser will have a UCSC genome browser, once we get to the point of having few enough contigs that building a browser makes sense.
- Style guide will have suggestions for organizing content on the wiki.
- Lecture notes will have pointers to notes taken by the students for each lecture. Two students will be designated each lecture to take notes, but others are encouraged to add their own notes to the wiki pages.
- Other mollusk genomes has a list of other mollusks that have been sequenced, and where the data for them is. Prior mollusk genomes

On the right side of the page, there is a "Table of Contents" sidebar with links to "Banana slug genomics", "Reborn - Spring 2015", "Important pages of this wiki", "Old pages of this wiki", and "Discussion". A vertical toolbar on the far right contains icons for search, clock, link, and an upward arrow.

Teams homepage

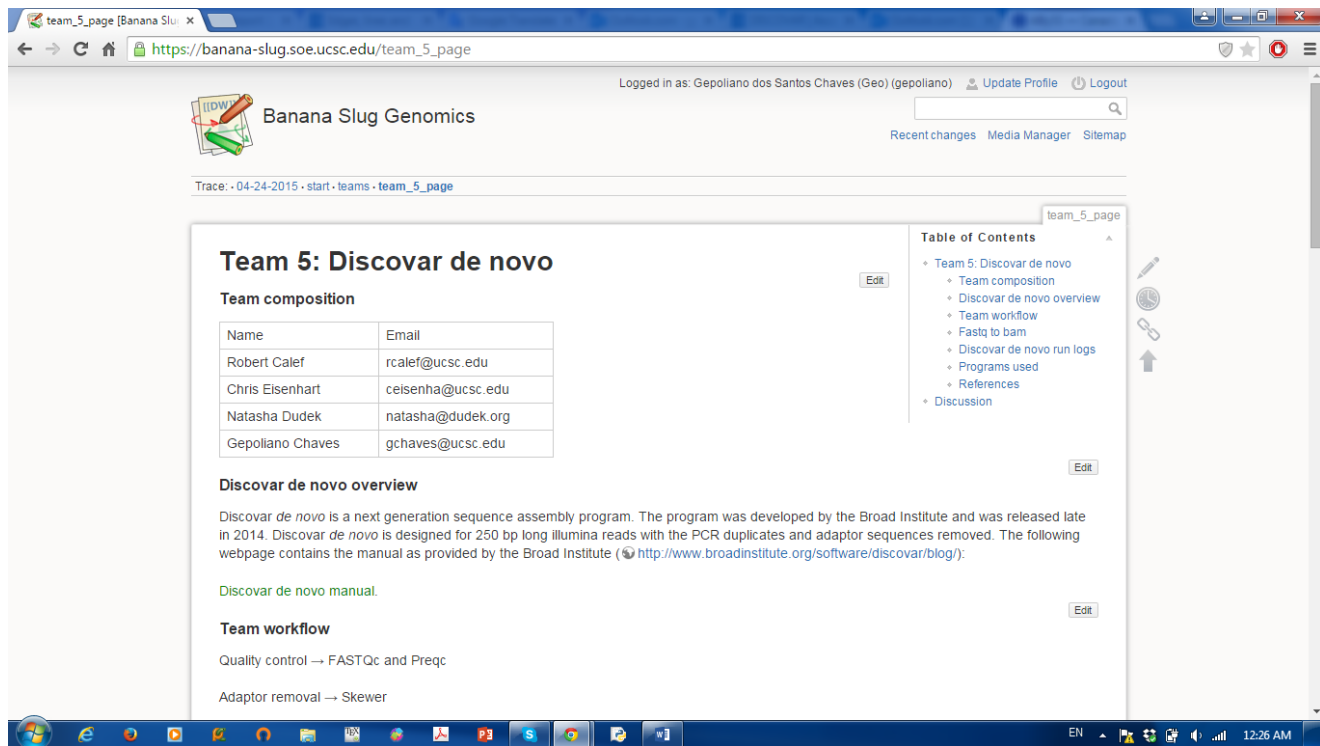


The screenshot shows a web browser window with the URL <https://banana-slug.soe.ucsc.edu/teams>. The page header includes the Banana Slug Genomics logo and a search bar. Below the header, there is a breadcrumb trail: `Trace: - team_5_page - start - teams`. The main content area is titled "Assembly Teams" and contains a table with the following data:

Number	Team members	Assembler	Documentation notes	Team page
1	Cole, McGovern, Houser, Richardson	Meraculous		
2	Matthew, Markello, Saremi	SOAP denovo	SOAPdenovo2	
3	Espinoza, Kan, Musselman-Brown	SGA	0.10.13	
4	Copher, Feal, Hussain	ABYSS		
5	Chaves, Calef, Dudek, Eisenhart	Discover denovo		Team 5

A black arrow points to the [Team 5](#) link in the 'Team page' column of the last row. Below the table, there is a note: "You could leave a comment if you were logged in." At the bottom of the page, there is a license notice: "Except where otherwise noted, content on this wiki is licensed under the following license: [CC Attribution-Noncommercial-Share Alike 3.0 Unported](#)". The footer also includes various icons and a timestamp: "teams.txt - Last modified: 2015/04/23 13:01 by ceisenhart".

Team 5 homepage: Discover de novo



The screenshot shows a web browser window displaying the 'team_5_page' on the Banana Slug Genomics website. The page is titled 'Team 5: Discover de novo' and includes a 'Team composition' table, a 'Discover de novo overview' section, and a 'Team workflow' section. A 'Table of Contents' sidebar is visible on the right.

team_5_page [Banana Slug] x

https://banana-slug.soe.ucsc.edu/team_5_page

Logged in as: Gepoliano dos Santos Chaves (Geo) (gepoliano) Update Profile Logout

Banana Slug Genomics

Recent changes Media Manager Sitemap

Trace: 04-24-2015 start teams team_5_page

Team 5: Discover de novo

Team composition

Name	Email
Robert Calef	rcalef@ucsc.edu
Chris Eisenhart	ceisenha@ucsc.edu
Natasha Dudek	natasha@dudek.org
Gepoliano Chaves	gchaves@ucsc.edu

Discover de novo overview

Discover *de novo* is a next generation sequence assembly program. The program was developed by the Broad Institute and was released late in 2014. Discover *de novo* is designed for 250 bp long illumina reads with the PCR duplicates and adaptor sequences removed. The following webpage contains the manual as provided by the Broad Institute (<http://www.broadinstitute.org/software/discovar/blog/>):

[Discover de novo manual.](#)

Team workflow

Quality control → FASTQc and Preqc

Adaptor removal → Skewer

Table of Contents

- Team 5: Discover de novo
 - Team composition
 - Discover de novo overview
 - Team workflow
 - Fastq to bam
 - Discover de novo run logs
 - Programs used
 - References
 - Discussion

Team composition

Team composition

Name	Email
Robert Calef	rcalef@ucsc.edu
Chris Eisenhart	ceisenha@ucsc.edu
Natasha Dudek	natasha@dudek.org
Gepoliano Chaves	gchaves@ucsc.edu

Team workflow

Team workflow

Quality control → FASTQc and Preqc

Adaptor removal → Skewer

PCR duplicate removal → FastUniq

Convert fastq files to BAM files → picard-fastqToSam

Discover *de novo* on data subsets → Discover *de novo* (with freq option)

Visualization of the output → ?

Edit

Discover *de novo* run logs

[Edit](#)

Discover run logs

Discover was designed for very specific data. To test the validity of our data we performed three different test runs. The test runs used a percentage of data from the three libraries. All the tests were run on .bam files on edser2. The run logs are stored as .txt files. The full logs can be seen on the wiki [here](#),

Run size	Data used
1% data	MiSeq data SW019_S1_L001, HiSeq data SW018_S1_L007, HiSeq data SW019_S2_L008
5% data	MiSeq data SW019_S1_L001, HiSeq data SW018_S1_L007, HiSeq data SW019_S2_L008
10% data	MiSeq data SW019_S1_L001, HiSeq data SW018_S1_L007, HiSeq data SW019_S2_L008

The logs are very large, important statistics have been gathered and are compared below.

	1% run	5% run	10 % run
Total runtime	1.75 hours	1.53 hours	2.4 hours
Peak memory use	43.92 GB	78.10 GB	151.05 GB
Bases in 1kb+ scaffolds	75,233	592,685	1,476,875
Bases in 10kb+ scaffolds	10,572	11,088	168,543

[Edit](#)

Discover *de novo* run logs

- Total runtime is relatively consistent
 - ReadQGrapher step scales with the data size
- RAM intensive steps
 - Reading in the files
 - ReadQGrapher (peak memory step)
- Ten fold increase in bases in 10kb+ scaffolds between 5% and 10% runs

Machine load

- Discover denovo reads all the data into RAM
 - roughly 2 bits of RAM per base
- CPU and RAM expensive for specific steps
 - Analyzing logs to narrow down which steps are expensive
 - ReadQGrapher

Program log

Programs used

The program, its location, and a brief, (brief!) explanation of what the program does

Picard

The Picard is a set of Java-based command-line utilities for SAM and BAM file manipulation (edser2:/soe/calef/picardtools and edser2:/soe/calef/picard_jars). Webpage: <http://picard.sourceforge.net/>.

Jemalloc

Is a general purpose malloc(3) implementation that emphasizes fragmentation avoidance and scalable concurrency support (edser2:/soe/calef/jemalloc). Webpage: <http://www.canonware.com/jemalloc/>.

Skewer

Skewer is an adapter trimmer for Illumina paired-end sequences (/campusdata/BME235/S15_assemblies/SOAPdenovo2/adaptorRemovalTask/skewer_run). Webpage: <http://sourceforge.net/projects/skewer/>.

FastUniq

FastUniq is a fast de novo duplicate removal tool for paired short DNA sequences (/campusdata/BME235/bin). Webpage: <http://sourceforge.net/projects/fastuniq/>.

GCC

GCC is a compiler for the GNU operating system. Webpage: <https://gcc.gnu.org/>.

Questions

- Pair correction and filling
- Sequence preparation
 - FastUniq
 - FastqToSam
- User experience
- Documentation and results

Acknowledgements

- Ed Green, sequencing and lectures
- Kevin Karplus, wiki, lectures, and past classes
- Stefan Prost, guest lecturer
- Steven Weber, Jared Copher, lab work
- John Pearse and Jan Leonard, slug biology
- Kickstarter donors, making it possible

Discover's origin and genetic variation

- At time previous to Discover's paper (Weisenfeld et al., 2014), the methods available to investigate GV did a good job in 90% of the cases.
- Calling variants was a challenging in the 10% remaining of the genome, specially those occurring in low-complexity sequence, segmental duplications and high GC content regions.

Discover's origin and genetic variation

- Discover was specifically designed to address challenging variant types;
- Discover involves initial alignment of reads to genomic regions followed by careful
- Compared with Genome Analysis Tool Kit (GATK), Discover provides a better coverage of challenging variants and excellent coverage of ordinary variants

Discover vs. Discover de novo

- Discover is a variant caller and small genome assembler
- Discover de novo however, can assemble genomes up to the mammalian size