This publication and its contents are proprietary to Illumina, Inc., and are intended solely for the contractual use of its customers and for no other purpose than to operate the system described herein. This publication and its contents shall not be used or distributed for any other purpose and/or otherwise communicated, disclosed, or reproduced in any way whatsoever without the prior written consent of Illumina, Inc.

For the proper operation of this system and/or all parts thereof, the instructions in this guide must be strictly and explicitly followed by experienced personnel. All of the contents of this guide must be fully read and understood prior to operating the system or any of the parts thereof.

FAILURE TO COMPLETELY READ AND FULLY UNDERSTAND AND FOLLOW ALL OF THE CONTENTS OF THIS GUIDE PRIOR TO OPERATING THIS SYSTEM, OR PARTS THEREOF, MAY RESULT IN DAMAGE TO THE EQUIPMENT, OR PARTS THEREOF, AND INJURY TO ANY PERSONS OPERATING THE SAME.

Illumina, Inc. does not assume any liability arising out of the application or use of any products, component parts, or software described herein. Illumina, Inc. further does not convey any license under its patent, trademark, copyright, or common-law rights nor the similar rights of others. Illumina, Inc. further reserves the right to make any changes in any processes, products, or parts thereof, described herein without notice. While every effort has been made to make this guide as complete and accurate as possible as of the publication date, no warranty or fitness is implied, nor does Illumina accept any liability for damages resulting from the information contained in this guide.

© 2008 Illumina, Inc. All rights reserved.

Illumina, Solexa, Array of Arrays, BeadArray, BeadXpress, CSPro, DASL, GoldenGate, Infinium, IntelliHyb, iSelect, Oligator, Sentrix, VeraCode, and Making Sense Out of Life are registered trademarks or trademarks of Illumina, Inc. Other brand and product names contained herein are the property of their respective owners.
# TABLE OF CONTENTS

**Chapter 1**  
Lab Workflow 1  
Technology Overview 2  
Data Analysis Workflow 4

**Chapter 2**  
Pipeline Workflow Overview 7  
Image Analysis 8  
Image Analysis Algorithm 8  
Image Analysis Offsets 8  
Base Calling 9  
Intensity Correction 10  
Phasing Correction 11  
Base Calling 12  
Preliminary Quality Score Estimate 13  
Alignment 13  
Quality Filtering 14  
Master Script 15  
Section Review 16

**Chapter 3**  
Run Folder Structure 19  
Preparing the Analysis 20  
GOAT Parameters 21  
Parameter Details 21  
GERALD Configuration Parameters 22  
ANALYSIS Options 23  
Reference Genome Options 23  
USE_BASES Option 24  
Mask Rules 24  
GERALD Lane Restrictions 25  
Example GERALD Configuration File 25  
Generating the Makefiles 25  
Pipeline Execution 26  
GOAT Sample Session 26  
Section Review 27
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Folder Structure</td>
<td>29</td>
</tr>
<tr>
<td>Pipeline Output</td>
<td>30</td>
</tr>
<tr>
<td>Run Statistics</td>
<td>30</td>
</tr>
<tr>
<td>Cluster Number</td>
<td>31</td>
</tr>
<tr>
<td>First Cycle Lane Intensity and Percent Intensity After 20 Cycles</td>
<td>32</td>
</tr>
<tr>
<td>Percent Passed Filters</td>
<td>33</td>
</tr>
<tr>
<td>Percent Aligned and Average Alignment Score</td>
<td>34</td>
</tr>
<tr>
<td>Error Rate</td>
<td>34</td>
</tr>
<tr>
<td>Errors.htm and Perfect.htm</td>
<td>35</td>
</tr>
<tr>
<td>Text Output Files</td>
<td>36</td>
</tr>
<tr>
<td>Final Result File: *_realign.txt</td>
<td>37</td>
</tr>
<tr>
<td>Final Result File: s_*_.sequence.txt</td>
<td>38</td>
</tr>
<tr>
<td>Final Result File: s_*_.export.txt</td>
<td>39</td>
</tr>
<tr>
<td>Section Review</td>
<td>39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise #1—Running the Pipeline</td>
<td>43</td>
</tr>
<tr>
<td>Exercise #2—Running a Second Alignment</td>
<td>44</td>
</tr>
<tr>
<td>Exercise #3—Skipping Cycles</td>
<td>49</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A—Installing the Data Set</td>
<td>53</td>
</tr>
<tr>
<td>Appendix B—Helpful UNIX Commands</td>
<td>54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Glossary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>55</td>
</tr>
</tbody>
</table>
INTRODUCTION

About the Course

The Genome Analyzer Pipeline course is a robust course designed for beginning users of Illumina’s Genome Analyzer Pipeline software. The intent of this course is to familiarize attendees who have little or no experience with the basic operation, analysis and common troubleshooting tips while working with Pipeline. The course provides tips and tools to attendees with intermediate knowledge to quickly be up and running the Pipeline software.

During this course, the following topics will be explored:

- Genome Analyzer Processes Overview
- Pipeline Components and Workflow
- Running the Pipeline
- Interpreting Pipeline’s Output
- Hands-on Exercises

Using the Student Guide

The Student Guide is your personal reference manual. Space is provided on each page for note taking.

Throughout this guide, tips on best practices and notes containing additional information or cautions are present. The icons listed below identify the kind of information being presented.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Note Tip Icon]</td>
<td>This icon indicates a note or tip.</td>
</tr>
<tr>
<td>![Warning Icon]</td>
<td>This icon indicates a warning.</td>
</tr>
<tr>
<td><strong>Text</strong></td>
<td>Text in this font indicates a UNIX command.</td>
</tr>
</tbody>
</table>

Student Guide Overview

Whether you downloaded this Student Guide before attending class or received it as a file transfer at the end of class, it serves as a valuable resource.

This document is designed to be printed double-sided and in color.
Genome Analyzer Overview
Lab Workflow

Prior to using the Pipeline software, a sample prep is created which is loaded on flowcell in the Cluster Station. Then the Genome Analyzer is utilized to generate the sequencing intensity data. The intensity data created by the Genome Analyzer is then analyzed using the Pipeline software.

Figure 1: Typical Lab Workflow Pre-pipeline
Technology Overview

DNA sequencing on the Genome Analyzer takes place in a specially designed device called the flow cell.

![Flow Cell Example](image)

**Figure 2: Flow Cell Example**

Flow cells contain eight discrete lanes, each of which can handle a separate DNA sample. Prepared DNA samples are loaded into the flow cell and clonally amplified to clusters using the Cluster Station.

![Simulated Lanes of a Flow Cell](image)

**Figure 3: Simulated Lanes of a Flow Cell**

During each sequencing cycle, the Genome Analyzer takes a series of images. Each imaged area is called a tile. Lanes are imaged in two or three columns of up to 110 tiles. Each tile should optimally contain 25,000 to 30,000 clusters.

A cluster is a clonal copy of a DNA fragment. During each sequencing cycle, a nucleotide is added to the template and the intensity from the incorporated fluorescently tagged nucleotides are imaged. Thus the cycle count is the same as the base count, and the terms are often used interchangeably. The sequence from a single cluster is often referred to as a “read.”

![Simulated Tiles on a Flow Cell](image)

**Figure 4: Simulated Tiles on a Flow Cell**
Each tile is imaged four times per cycle (one image per base). Depending upon the number of sequencing cycles, a single sequencing run can produce upwards of 350,000 images. At 2.5 - 3.0 Mb for each image, data transfer and storage becomes an important consideration.

*Figure 5: Simulated Image of a Lane and an Image*
Data Analysis Workflow

The data analysis workflow diagram illustrates how the programs work together. Images are transferred from the Genome Analyzer to the Analysis server and input into the Analysis Pipeline. The Analysis Pipeline performs three major steps: image analysis, base calling, and alignment Analysis.

**Figure 6: Example of Data Analysis Workflow**

The IPAR (Integrated Primary Analysis and Reporting) system performs image analysis during the Genome Analyzer run. The intensity files computed by IPAR are the same as those computed by the pipeline can be used as an alternative input into the pipeline. This can result in major time savings.

**Figure 7: Data Analysis Workflow Using IPAR**
CHAPTER 2

Pipeline Components and Workflow
Pipeline Workflow Overview

The Pipeline software is comprised of four main parts:

1. Master script (General Oligo Analysis Tool [GOAT])
2. Image analysis (Firecrest)
3. Base calling (Bustard)
4. Alignment (Generation of Recursive Analyses Linked by Dependency [GERALD])

Figure 8: Pipeline Workflow Overview

Note: The Glossary section of this Student Guide has terms and acronyms defined.
Image Analysis

Firecrest is the name given to the portion of the Pipeline software that performs image analysis on the TIFF files generated by the Genome Analyzer. The main function of Firecrest is to identify clusters and extract intensities from the TIFF images.

![Image Analysis Algorithm Example](image)

**Figure 9: Image Analysis Algorithm Example**

Image Analysis Algorithm

Firecrest creates the intensity files by the following three part algorithm:

1. A band-pass filter locates, defines, and sharpens the cluster’s borders;
2. A threshold determines the amount of background noise to differentiate the maximum value of the cluster from the surrounding noise; and
3. Maxima detection finds the cluster(s) and quantifies their intensities.

Image Analysis Offsets

Image analysis offsets are required to correct the scale and the registration of the images for each fluorophore.

![Image Analysis Offsets Example](image)

**Figure 10: Image Analysis Offsets Example**
Each of the four images are slightly different, as the optical path through a prism varies slightly for different wavelengths of light. The clusters are aligned in each quadrant of the image individually, then again for the image as a whole as illustrated by the yellow cross-hair.

Offsets are computed over all cycles and stored in a file labeled default_offsets.txt. Table 1 displays a sample offsets file. The first two columns show how to shift the image in the X and Y direction. The last two columns show how much to shrink or stretch the image.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.32</td>
<td>1.41</td>
<td>0.00069</td>
<td>0.00068</td>
</tr>
<tr>
<td>-0.01</td>
<td>1.82</td>
<td>-0.00123</td>
<td>-0.00125</td>
</tr>
<tr>
<td>0.14</td>
<td>1.59</td>
<td>-0.00097</td>
<td>-0.00092</td>
</tr>
</tbody>
</table>

Table 1: default_offsets.txt File Example

Base Calling

Base calling is performed by Bustard, the second section of the Analysis Pipeline. During base calling, Bustard takes the raw intensity files as input and provides cluster sequences and preliminary quality scores.

Bustard performs base calling in four steps:

1. Intensity correction
2. Phasing correction
3. Base determination
4. Preliminary quality score estimation
Intensity Correction

There are two components of intensity correction:

1. **Cross talk**
   - The Genome Analyzer uses four dyes that are excited by two lasers. With multi-dye systems, there is a certain amount of spectral cross talk between the dyes.
   - Cross talk correction is a process to correct for the frequency overlap between the dyes.
   - Cross talk is illustrated in Figure 11. The frequency response curves of two dyes are shown. At the maximum excitation frequency of dye Y, there is a small amount of fluorescence from dye X. Similarly there is a certain amount of fluorescence from dye Y at the maximum peak of dye X.

2. **Normalization**
   - Scales intensities so the magnitudes are equivalent.
   - The dyes are not equally efficient at fluorescing.

*Note*  Cross talk is not reciprocal; the amount of Y in X is not the same as the amount of X in Y.

*Figure 11: Cross Talk Illustration Example*
The cross talk and the normalization corrections are encapsulated in the “matrix.txt” file. This file represents both the adjustments for cross talk and dye normalization. An example is shown in Figure 12. The cross talk matrix is computed based on cycle 2. The column and row order are given by the header in Figure 12.

The columns represent the nucleotide being measured, and the rows represent the contribution from the other nucleotides. The diagonal represents the scaling needed for normalization.

```
#  Auto-generated frequency response matrix
> A
> C
> G
> T
0.65 0.14 0.01 0.02
0.81 1.04 0.02 0.03
0.02 0.02 1.18 0.04
0.03 0.03 1.10 1.56
```

*Figure 12: matrix.txt File Example*

**Phasing Correction**

Phasing correction is an important step needed to correct for minor errors in the synthetic chemistry. This is achieved through:

- **Phasing**
  - In any given cycle, a few DNA molecules within a cluster may fail to extend. This is referred to as “phasing,” as the molecules which fail to extend are now out of phase with the rest of the cluster. Phasing can occur when a base is not incorporated, or a nucleotide does not deblock from the previous cycle.

- **Prephasing**
  - A second type of chemistry error is known as “prephasing.” Prephasing occurs when an unblocked base is incorporated in the growing DNA molecule, allowing a second nucleotide to be incorporated. The resultant strand is then a base ahead.
Phasing is a chemistry issue and is highly dependent upon the experimental conditions. Thus, the phasing and prephasing contributions need to be estimated for each run. In order to accurately estimate the phasing, at least one lane on the flow cell must contain a sample with a random, balanced base composition. This is often best achieved by the use of a PhiX control lane.

Base Calling

After all the corrections are applied to the intensities, the raw base is determined/called. The base with highest corrected intensity is called. In Figure 14, “C” is the highest called base.

In the case of a tie, the bases are called in alphabetical order. When this occurs, the actual call is typically irrelevant as the quality filters will reject that base.
Preliminary Quality Score Estimate

Independent of the base call, an initial quality score is estimated. The initial score is based on several criteria, including the intensity and the background noise. A quality score is estimated for each base. These scores are un-calibrated, and the estimate may not be completely accurate.

Alignment

The third section of the Analysis Pipeline performs genomic alignments and outputs the final results files. This section is referred to as Generation of Recursive Analyses Linked by Dependency (GERALD).

The sequences are first aligned to the reference sequence, then filtered for problematic clusters. The resulting alignments and sequence statistics are then summarized in a collection of text and HTML reports.

GERALD supports analysis with two alignment algorithms: PhageAlign and Efficient Large-Scale Alignment of Nucleotide Databases (ELAND). Both algorithms are non-gapping and produce output in similar formats.

1. PhageAlign
   - Is a useful algorithm for viral genomes and Bacterial Artificial Chromosomes (BACs). Due to that fact that it is slow and memory hungry, PhageAlign should not be used for larger genomes. PhageAlign is a greedy algorithm, in that it always finds a match no matter how poor, so the results require more indirect analysis.

2. ELAND
   - A proprietary algorithm that greatly speeds alignments of small reads. The first 32 bases of the sequence are used to create a seed alignment. Within the seed alignment there can be no more than two mismatches; sequences with more than two mismatches are marked “NM” for no match.
   - For sequences longer than 32 bases, the best seed alignments are extended for the length of the read and then scored. The best scoring alignment is reported.
Quality Filtering

Quality filtering removes sequences with low quality base calls. The default filter is CHASTITY. The CHASTITY equation divides the intensity of the highest peak by the sum of the two highest intensities.

\[ C = \frac{I_A}{I_A + I_B} \]

Quality filters are applied across a specific number of bases. The default number is 12, but can be changed at the user’s discretion. By default, filters are applied across the first 12 cycles. This can be altered by the PURE_BASES setting in the GERALD configuration file.
Master Script

The last section of the Analysis Pipeline is a master script that creates the files and directory structures needed by the other pipeline programs. This script coordinates the running of the entire pipeline. The master script is referred to as General Oligo Analysis Tool (GOAT). The GOAT script only sets up the analysis directories and creates the makefiles.

Makefiles are a programming tool that define dependencies between sets of files and rules for keeping the files up-to-date. Makefiles are interpreted by the utility program “make” which follows the rules and runs the files.

Figure 16: Master Script File Structure Illustration
Section Review

Image Analysis
  • Offset correction
  • Cluster extraction and quantification
  • Referred to as Firecrest

Base Calling
  • Normalization matrix
  • Phasing/prephasing correction
  • Referred to as Bustard

Alignment
  • PhageAlign or ELAND algorithms
  • Referred to as GERALD

Master Script
  • Sets up data folders and creates makefiles
  • Referred to as GOAT
Running the Pipeline
Run Folder Structure

When the images are copied from the Genome Analyzer, they are placed in the experiment directory (sometimes called the "run folder"). Pipeline depends on a specific "run folder" structure, as shown in Figure 17. The "run folder" contains information about the date, the Genome Analyzer, and the flow cell. Inside the experiment directory are subdirectories containing the images and other files generated during the Genome Analyzer run.

When the GOAT script is executed, it sets up the data directory and the rest of the analysis directory structure. The analysis directories have date and time stamps incorporated into the name so older directories containing results from previous analyses are not overwritten.

The analysis directory structure is nested, just like the GOAT/Bustard/GERALD analyses. The final output is found in the GERALD directory.

Figure 17: Run Folder Structure Example
Preparing the Analysis

The overall command to run the GOAT script is shown in Figure 18.

```bash
<PipelinePath>/Goat/goat_pipeline.py
    [--cycles=<firstcycle>-<lastcycle>]
    [--tiles=s_<lane>[_<tile>][, s_<lane>[_<tile>]]
    [--offsets=<offsetfile>]
    [--matrix=<matrixfile>|auto<lane>]
    [--phasing=<phasingratio>|auto<lane>]
    [--prephasing=<prephasingratio>]
    [--control-lane=<lane>]
    [--GERALD=<configfile>]
    [--make]
    <run-folder>|<IPAR-folder> [<run-folder-2>]
```

*Figure 18: GOAT Command Script and Options Example*

The command is written using standard UNIX conventions. Portions of the command enclosed in angle brackets "<" and ">" are place-holders to be replaced with specific values.

For example "<PipelinePath>" should be replaced with the actual path to the pipeline installation. Commands in square brackets "[" and "]" are optional arguments and should be used only if needed.
GOAT Parameters

The GOAT parameters can be broken down into the same categories as the analysis program. There are parameters that control the image analysis, base calling, and the alignment steps. GOAT’s parameters and the permissible options:

**Image Analysis**

--cycles  Specify the range of cycles to analyze

--tiles  Specify specific tiles to analyze

--offsets  Offsets file with relative or absolute path; auto will prompt the pipeline to compute the offsets

**Base Calling**

--matrix  Specify matrix file for base calling; auto will prompt the pipeline to compute the matrix; auto<n> will only use lane <n> to compute the offsets

--phasing  specify rate of phasing auto will prompt pipeline to compute auto<n> will only use lane <n> to compute phasing

--prephasing  Specify rate of pre-phasing

**Alignment**

--GERALD  Configuration file for alignment

**Parameter Details**

cycles
The cycles are a range of cycles to analyze (e.g., 2-17). Be careful when interpreting analyses with this parameter: the pipeline will renumber the cycles beginning with one.

Syntax Example: [--cycles=1-36]

tiles
The tiles parameter is an advanced option. Tiles are specified by s_<lane>_ <tile> and can only be included; thus this parameter is cumbersome if you want to exclude one or two tiles.

Syntax Example: [--tiles=s_1, s_2_0014]

offsets
As of Pipeline v0.3 the offsets parameter can be safely ignored.

Syntax Example: [--offsets=offsets.txt]
**matrix**
The matrix parameter controls how the cross talk matrix is generated. Setting [--matrix=auto] is the default; this uses the entire flow cell to compute the matrix. If all the samples on the flow cell are not appropriate for matrix calculation, restrict it to a control lane using the lane number in place of <n>. Finally, matrix can take a filename; this is an emergency measure only to allow the use of a matrix file from a previous run in the event control lane fails.

Syntax Example: [--matrix=auto5]
Syntax Example: [--matrix=matrix.txt]

**phasing**
The phasing parameter is analogous to the matrix parameter: it allows control over the computation of the phasing and prephasing values. Setting phasing=auto is the default and uses the entire flow cell. Phasing can use a value from a previous run; this is an emergency measure only in case the control lane failed.

Syntax Example: [--phasing=auto]
Syntax Example: [--phasing=0.008]
Syntax Example: [--prephasing=0.0055]

**Warning**
In the Summary.htm page the value is presented as a percentage, which will need to be converted to a decimal.

**control-lane**
This parameter replaces phasing=auto<n> and matrix=auto<n>

Syntax Example: [--control-lane=5]

**GERALD**
Unlike image analysis and base calling, there are too many potential parameters for alignment to specify them all on the command line. Thus, the GERALD command specifies the file path and name of a configuration file for the alignment module. If the GERALD configuration file is not specified, the GOAT command sets up only the Firecrest and Bustard directories

**GERALD Configuration Parameters**
The basic purpose of the GERALD configuration file is to control what and how analyses are done. There are parameters which control the type of alignment, the reference genome files, and which bases to use. There are also options which control e-mail notifications for when a run is complete, and controls for post-pipeline commands.
ANALYSIS Options

There are six analysis options for the Pipeline:

<table>
<thead>
<tr>
<th>Analysis Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYSIS eland</td>
<td>Original ELAND alignment</td>
</tr>
<tr>
<td>ANALYSIS eland_extended</td>
<td>ELAND alignment with improved long read support</td>
</tr>
<tr>
<td>ANALYSIS eland_pair</td>
<td>ELAND alignment with support for paired-end reads</td>
</tr>
<tr>
<td>ANALYSIS sequence</td>
<td>Output filtered sequence only</td>
</tr>
<tr>
<td>ANALYSIS none</td>
<td>Do not perform any analysis</td>
</tr>
<tr>
<td>ANALYSIS default</td>
<td>PhageAlign alignment</td>
</tr>
</tbody>
</table>

Table 2: Analysis Options for the Pipeline

In general, “ANALYSIS eland_extended” is the best choice for single read sequence alignments. “ANALYSIS eland_pair” is the best choice for paired-end read alignments, as it performs post-alignment read pairing. If no alignment is desired, then “ANALYSIS sequence” creates the filtered sequence output.

“ANALYSIS default” performs alignments using PhageAlign. There are also some older analysis options, namely “ANALYSIS expression” and “ANALYSIS monotemplate.” These PhageAlign based analysis options still exist to maintain backwards compatibility, but they are not recommended.

Reference Genome Options

PhageAlign and ELAND, the different alignment programs, use slightly different files for alignment.

- PhageAlign (GENOME_FILE)
  - PhageAlign uses a single FASTA file for the reference sequence. Note that the file must contain only a single entry. The FASTA file is specified by the GENOME_DIR and the GENOME_FILE parameters.
  - The value for the GENOME_DIR parameter is the path to the directory which contains the FASTA file, while the value of the GENOME_FILE is the name of the FASTA file.

- ELAND (ELAND_GENOME)
  - The ELAND program uses a specially prepared set of reference files. These compressed files are created using the squashgenome program. The squashgenome program compress any FASTA file, including FASTA libraries (files with more than one sequence entry).
  - The ELAND_GENOME parameter is the path to a directory containing a set of compressed reference sequences. ELAND searches all the compressed files in the directory.
USE_BASES Option

The USE_BASES parameter specifies which bases to use in the alignment. The value of the USE_BASES is a character string known as a mask, which turns bases on and off. In the simplest form, we can use all bases in the query sequences by placing “USE_BASES all” into the configuration file. This tells the Analysis Pipeline to generate a mask which uses all the bases.

To have more control over the bases used in the alignment, the “USE_BASES <mask>” parameter is supplied, where the “<mask>” is a replaced expression.

Warning

Even though it is possible, don’t drop bases in the middle. There are no place-holders in the sequence for skipped bases, so the sequence on either side of the dropped base will be concatenated. In essence, this would create an artificial indel, and the alignments would all fail.

Mask Rules

The rules for building the mask are:

- ‘n’-Means ignore the cycle
- ‘Y’-Means use the cycle for the alignment
- ‘,’ (comma)-Denotes a read boundary (for multiple reads)
- ‘*’ (asterisk)-Means “fill up the read as far as possible with the preceding character” (as in a UNIX regular expression)
- Any number-Means that the previous character is to be repeated as many times

<table>
<thead>
<tr>
<th>Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE_BASES Y35</td>
<td>Single 35-base read</td>
</tr>
<tr>
<td>USE_BASES nY34</td>
<td>Single 35-mer read, first base masked</td>
</tr>
<tr>
<td>USE_BASES nY33n</td>
<td>Single 35-mer read, first and last base masked</td>
</tr>
<tr>
<td>USE_BASES nY*n</td>
<td>Single 35-mer read, first and last base masked</td>
</tr>
<tr>
<td>USE_BASES Y35,Y35</td>
<td>Paired 35-base reads</td>
</tr>
</tbody>
</table>

Table 3: USE_BASES Mask Examples
GERALD Lane Restrictions

Application of configuration parameters to specific lanes is achieved by specifying the lane(s) at the left of the command followed by a colon. For example:

```
123:ANALYSIS eland
567:ANALYSIS sequence
8:ANALYSIS none
```

Parameters without lanes specified applies to all lanes. Lanes not specified for a parameter uses default values.

Example GERALD Configuration File

```
GENOME_DIR /data/GENOMES
4:GENOME_FILE phix.fa
1235678:ANALYSIS eland
123:ELAND_GENOME /data/GENOMES/ELAND/human
56:ELAND_GENOME /data/GENOMES/ELAND/mouse
78:ELAND_GENOME /data/GENOMES/ELAND/rat
USE_BASES all
EMAIL_SERVER mail_server1
EMAIL_DOMAIN someplace.com
EMAIL_LIST somebody@someplace.com
WEB_DIR_ROOT http://data/SLXAUS-EMC
```

*Figure 19: Example of a matrix.txt Configuration File*

Generating the Makefiles

Providing the --make parameter to the GOAT command is required in order to generate the files and directories. Running GOAT without --make performs a "sanity check" to ensure all required files are present.

In Figure 19, the user placed the PhiX control in Lane 4 and the samples in the other lanes. PhageAlign is used to align against the phix_plus_snps.fa file, so there is no need to specify default analysis option, although it would make it clearer as to what is being done. The ELAND program is used for all other lanes. The ELAND_GENOME specifies that lanes 1, 2, and 3 are human, lanes 5 and 6 are mouse, and lanes 7 and 8 are rat. The USE_BASES specifies that all the bases in the reads are used.

The remainder of the parameters specify e-mail notification at the end of the run.
Pipeline Execution

After the GOAT command is used to create all the directories and files, the make command runs the pipeline.

Type make with the process command:
\[-j<N>\]: Run parallel analysis on a multiprocessor system
\(<N>\) is the number of processors to be used

And the target:
\[\text{recursive}: \text{ Runs analysis in the current directory and in all available subdirectories}\]

The software performs a complete end-to-end analysis run from image analysis to alignment. Note, this version of the command is only for single servers -- clusters have a more complex invocation.

GOAT Sample Session

1. Test command.
   
   \[\text{goat_pipeline.py --GERALD=\textnormal{config.txt}.}\]

2. Create makefiles.
   
   \[\text{goat_pipeline.py --GERALD=\textnormal{config.txt}. --make}\]

3. cd into the analysis directory (It is important to switch to the specific analysis folder to actually run the make executable. For a complete analysis run, the makefile to execute will be in Firecrest directory.)

4. Execute makefiles (It is important to remember the --recursive flag to trigger the complete end-to-end analysis; without recursive only the current module is run.)
   
   \[\text{make } -j8 \text{ recursive}\]

The analysis takes a while to run. While the actual run-time is dependent upon the hardware, analysis can take 36 hours or more. Use of the UNIX nohup command serves to prevent interruption of the pipeline and to capture the output from the software.
Section Review

**Run Folder Structure**
- Nested folder structure
- Setup by GOAT

**GOAT Options**
- Tiles (specify tiles or lanes to analyze)
- Cycles (range of cycles to analyze)
- Alignment options set in separate GERALD config.txt file

**GERALD Options**
- ANALYSIS eland extended
- ANALYSIS eland_pair for paired-end reads
- ANALYSIS sequence filtered sequence without alignment
- ANALYSIS none not analyzed

**Reference Genome**
- ELAND Genome must be squashed from single entry FASTA file
- Use Bases masking beginning or ends of reads from alignment

**Execute by make**
- `-- make` option required to execute script
- `-j<N>` specifies number of processors
- Recursive option to run full Pipeline
Interpreting the Output
Run Folder Structure

Each analysis run creates a set of folders. These folders are named with the run details and dates to keep them unique. Note that the date stamp is the date the GOAT command was run, not the date the make command was run. If the goat_pipeline.py is invoked multiple times on the same day, the directories are further distinguished with a version number.

![Image](image.png)

*Figure 20: Run Folder Structure Example*

It is possible to run portions of the analysis without running other parts. To do so, one only needs to invoke the proper sub-script (e.g., Bustard.py or GERALD.pl). In the example above, the Pipeline was first run completely to create the directory structure under “Image Analysis 1.” The image analysis and the base calling were then run a second time, to create the directories “Image Analysis 2” and “Base calling 2a.” There was no --GERALD argument provided to the GOAT command, so there is no Gerald directory under 2a.

The Bustard script was run again by itself, creating the “Base calling 2b” directory. The Gerald script was then run twice on the sequences in the 2b directory, creating the two different sequence alignment folders.

Pipeline Output

The final output files are found in the GERALD directory. There are two types of output:

1. HTML summary files which are viewable in any web browser
2. Text-based report and alignment files

There are also numerous intermediate files which allow the Pipeline to be run multiple times at any stage.
Run Statistics

Comprehensive results and performance measures are saved in Summary.htm.

Summary.htm provides an overview of the results which allows for a quick assessment of the results. Two tables in Summary.htm summarize experiment performance: Lane Results Summary and the Expanded Lane Summary.

### Lane Results Summary

<table>
<thead>
<tr>
<th>Lane</th>
<th>Lane Yield (kloosters)</th>
<th>Clusters (raw)</th>
<th>Chosters (PF)</th>
<th>1st Cycle Int (PF)</th>
<th>% intensity after 20 cycles (PF)</th>
<th>% PF Clusters</th>
<th>% Align (PF)</th>
<th>Alignment Score (PF)</th>
<th>% Error Rate (PF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5934</td>
<td>16577 +/- 655</td>
<td>12337 +/- 454</td>
<td>2185 +/- 48</td>
<td>89.49 +/- 2.04</td>
<td>74.43 +/- 0.70</td>
<td>65.72</td>
<td>11237.57 +/- 92.87</td>
<td>3.61 +/- 0.06</td>
</tr>
</tbody>
</table>

Tile mean across clap

AV. 16577 12337 2185 89.49 74.43 65.72 11237.57 3.61

### Expanded Lane Summary

<table>
<thead>
<tr>
<th>Lane</th>
<th>Clusters (tile mean) (raw)</th>
<th>% Phasing</th>
<th>% Prephasing</th>
<th>% Error Rate (raw)</th>
<th>% Error Rate (tile mean)</th>
<th>% Equivalent Perfect Clusters (raw)</th>
<th>% Equivalent Perfect Clusters (tile mean)</th>
<th>Cycle 2-4 Avg Int (PF)</th>
<th>Cycle 2-10 Avg % Loss (PF)</th>
<th>Cycle 19-20 Avg % Loss (PF)</th>
<th>% Align (PF)</th>
<th>% Error Rate (PF)</th>
<th>% Equivalent Perfect Clusters (PF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16577</td>
<td>0.6800</td>
<td>0.3500</td>
<td>3.81</td>
<td>74.43</td>
<td>2070 +/- 60</td>
<td>0.65 +/- 0.13</td>
<td>0.10 +/- 0.09</td>
<td>0.10 +/- 0.09</td>
<td>65.72</td>
<td>3.61</td>
<td>7525</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 21: Lane Results and Expanded Lane Summary File Example**

In Figure 21, the Lane Results Summary table shows the standard run metrics. For each lane, there are seven summary values. The "Clusters" column shows the average number of clusters per tile in the lane, along with the standard deviation. The "Av 1st Cycle Int" is the average for all clusters. The "%intensity after 20 cycles" column is also the average for all clusters. "%PF Clusters" is the percentage of clusters that passed the quality filters.
Cluster Number

Cluster number is how many clusters are found in an average tile. For low-density flow cells there should be up to 6,000 clusters, while for high-density flow cells there are between 20,000–35,000 clusters. The number of clusters should not be used as a performance criteria; it is provided to enable yield calculation and to provide a check that the sample dilution is correct. However, it is useful to remember that the phasing and matrix computations are stochastic, so a very low number of clusters (i.e., less than 1,000 clusters per tile) can lead to invalid results.

It is also good to remember that a high number of clusters is self-defeating. With few clusters on the surface, then any new cluster will most likely fall into “empty space” and generate a new output sequence. As more and more clusters get added, the probability of one cluster falling on top of another cluster (within a certain distance) increases. Finally an optimal density is reached, and beyond that any additional cluster is more likely to produce another “overlap” or “collision” than to fall into empty space.

Clusters that fall into “collisions” (again, within a certain radius) contain mixed signals and become useless for base calling and sequencing.

Figure 22 illustrates the relationship between cluster size and density:
- The X-axis shows the number of actual clusters on the flow cell.
- The Y-axis shows the number of detectable clusters that pass the quality filter, and the curves show the relationship for different sized clusters.
- The clusters generated by the current Cluster Station protocol are between 25 and 30 square pixels, shown by the red and blue curves respectively, and it shows the optimal cluster density is approximately 30,000. Denser clusters reduce the number of clusters which pass the filters.

![Figure 22: Number of Clusters vs. Clusters Detected File Example](image-url)
First Cycle Lane Intensity and Percent Intensity After 20 Cycles

The First Cycle Intensity is the average first cycle intensity of all color channels, in all tiles in the lane. Very low initial intensities (typically defined as less than 800) can disrupt the analysis, as the clusters will be indistinguishable from noise in later cycles. Percent intensity after 20 cycles should be 50% or more.

These criteria should be interpreted together to judge the run quality. There is a concern about how much signal is available in later cycles. Short runs may be okay with lower intensities, as long as there is little intensity loss. On the other hand, a high first cycle intensity is worthless if there are heavy intensity losses over early cycles.

The intensities are graphically displayed in the Intensity Versus Concentration (IVC) plots. IVC plots are accessible through a link below the summary tables. There are five different IVC plots for each lane, three of which are shown in the Figure 23.

The “All” plots show the average intensity of clusters in the four channels, irrespective of base call. That is, for all clusters the intensities in all four images are averaged. The relative intensities correspond to the base composition of the sample. This plot should show a slight decay curve, and the intensity variation from cycle to cycle to be minor. Cycles with large spikes or drops in intensity can be indicative of cycle to cycle focus or fluidics problems, and there will be a higher error rate at that base position.

Not shown are the “% All” and the “% Called” plots. These contain the same data as “All” and “Called” but are normalized to remove intensity drift.
The “Called” plots show the average intensity of clusters in the four channels for the called bases. That is, for all the clusters the average intensity of the channel corresponding to the called base is averaged, and the three intensities corresponding to the uncalled bases are ignored. Thus the “called” intensities should be approximately three to four times the “all” intensities. Typically for genomic samples where all bases are present, all four channels should have roughly equal intensity. As with the “All” plot, there should be a smooth decay curve with an obvious downward slope, and the intensity variation from cycle to cycle to be minor. Deviations might indicate poor fluidics or a poorly blocked flow cell, although if present from cycle 1 the initial matrix estimate may be in error.

The third plot “% Base Calls” should represent the base composition of the sample. For random libraries (e.g. genomic samples) one would expect the four bases to be approximately equal. Further, for random genomic libraries each base should be a horizontal line, and deviations indicate miscall issues that will affect the error rate. For non-genomic samples where the base composition is not random, this plot is not useful.

**Percent Passed Filters**

The quality filters pass only sequences that meet the quality filter criteria. The CHASTITY filter ensures the intensity of the called base is greater than that of the next highest intensity. CHASTITY specifically filters merged clusters, although clusters with odd shapes and clusters too close to the edge of the tile are also filtered.

For flow cells with 20,000–35,000 clusters the percentage of clusters passing filtering might be between 40% and 50%. However, this value should not be treated as a performance criterion, because sequences that make it past the filter are good quality. It is provided only to help compute yield.

**Percent Aligned and Average Alignment Score**

The percent aligned and average alignment scores are computed using only sequences that pass the filter criteria. In addition, only unique hits are included in the calculation.

Scoring is based upon summing the individual base scores, which are +500 for a match and -547 for a mismatch when using ELAND or PhageAlign. If eland_extended is used the scores are based on a Phred-like score.

The expected values for these metrics depends upon the genome. For PhiX controls and well characterized genomes, the percent alignment should be nearly 100%. The percent aligned has different interpretations for PhageAlign and ELAND.

PhageAlign always aligns sequences to nearly 100%, despite the scores. In contrast, because ELAND limits alignments to two mismatches it always reports good scores despite the percentage aligned. Thus, if there is an alignment problem; PhageAlign has high percent aligned and low scores, while ELAND has low percent aligned and high scores.
**Error Rate**

This column reports the average error rate of all sequences that have passed the filter. The current metric for runs is an error rate less than 1.5% for 25 aligned bases using PhageAlign.

Errors.htm shows overall error rates for all lanes by cycle. The overall error rate should be less than 1% over early cycles, gradually increasing in later cycles. The error rate represents the average miscall rate based on alignment with a known sequence. It should show consistency from tile to tile within a lane, and from lane to lane if the same sample was used in the lanes.
Errors.htm and Perfect.htm

Error rate variability between tiles may be caused by a number of experimental issues, including bubbles, rapid focus fluctuations, contamination on the flow cell exterior (i.e. dirt, fingerprints, fluid, oil).

File Error rates eventually rise at high read-lengths. Error rates can rise more rapidly than normal due to issues such as low intensity at start, high decay rate, and high phasing or prephasing. High and constant error rates from cycle1 may mean genomic contamination or an incorrect reference.

Figure 24: Errors.htm File Example
Text Output Files

The pipeline produces a lot of files. Most are considered intermediate files, and should be used with caution.

<table>
<thead>
<tr>
<th>Image Analysis</th>
<th>Base Calling</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>- *.int.txt</td>
<td>- *.prb.txt</td>
<td>- *.eland.txt</td>
</tr>
<tr>
<td>- *.nse.txt</td>
<td>- *.qhg.txt</td>
<td>- *.align.txt</td>
</tr>
<tr>
<td></td>
<td>- *.seq</td>
<td>- *.prealign.txt</td>
</tr>
<tr>
<td></td>
<td>- *.sig2.txt</td>
<td>- *.score.txt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- *.prescore.txt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- *.qraw.txt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- *.qval.txt</td>
</tr>
</tbody>
</table>

Only specific files from the alignment step are considered final output files.

Alignment

- *.realign.txt (ELAND)
- *.rescore.txt (ELAND)
- *.sequence.txt
- *.qcal.txt
- *.qcalreport.txt
- *.export.txt (ELAND Extended)
- *.anomoly.txt (ELAND Pairs)
**Final Result File: ***_realign.txt*

The realign file is the standard output file for both PhageAlign and ELAND, but not if ELAND is run with the eland_extended option. This file contains only those sequences which have passed the filter. Columns 1, 2, and 3 appear for every sequence:

- Column 1 is the sequence
- Column 2 is the best alignment score for the sequence, and
- Column 3 is the number of places the sequence aligned.
- Columns 4 through 7 appear only if there is a single unique alignment.
- Column 4 gives the reference name and the position,
- Column 5 gives the strand orientation.
- The sequence of the reference is given in Column 6.
- Column 7 gives the score of the next best alignment.

**Final Result File: s_*_sequence.txt**

This file contains the quality filtered output sequences. These sequences have passed the quality filtering metrics and have aligned to the genome. The format of the file is controlled by the GERALD configuration file. The parameter SEQUENCE_FORMAT has three possible options:

- SEQUENCE_FORMAT --fasta
- SEQUENCE_FORMAT --fastq
- SEQUENCE_FORMAT --scarf
Final Result File: s_*_export.txt

The export file contains entries for all sequences in a lane regardless of their filter status. There is one sequence per line, and the fields are tab separated. Thus it can be imported directly into a database. This file is produced by the eland_extended analysis mode, and replaces the *_realign.txt file.

The number of fields per line remains a constant 22, and any fields not relevant to a particular read are left blank (the empty string ""). In particular, for a single-read analysis the Read Number, Paired Read Alignment Score and Partner Chromosome/Contig/Offset/Strand fields will all be blank. The fields are:

1. Machine (as parsed from run folder name).
2. Run Number (as parsed from run folder name).
3. Lane.
4. Tile.
5. X Coordinate of cluster.
6. Y Coordinate of cluster.
7. Index String (blank for a non-indexed run).
8. Read Number ("1" or "2" for paired read, blank for a single read).
9. Read.
10. Quality String-In symbolic ASCII format (ASCII character code = quality value + 64) by default, set QUALITY_FORMAT --numeric in the GERALD configuration file to get numeric values instead.
11. Match Chromosome-Name of chromosome which the match was to or code indicating why no match was done.
12. Match Contig (blank if no match found)-Gives contig name if there is a match and the match chromosome is split into contigs.

13. Match Position (always with respect to forward strand, numbering starts at 1).

14. Match Strand ("F" for forward or "R" for reverse; blank if no match).

15. Match Descriptor-Concise description of alignment. A numeral denotes a run of matching bases; a letter denotes substitution of a nucleotide (e.g. a 35 base read, “35” denotes an exact match and “32C2” denotes substitution of a “C” at the 33rd position).

16. Single Read Alignment Score-Alignment score of single read match (if a paired read, gives alignment score of read if it were to be treated as a single read).

17. Paired Read Alignment Score-Alignment score of read pair (alignment score of a paired read and its partner, taken as a pair. Blank for a single read run).

18. Partner Chromosome-Not blank only if read is paired and its partner aligns to another chromosome, in which case it gives the name of the chromosome.

19. Partner Contig-Not blank only if read is paired and its partner aligns to another chromosome and that partner is split into contigs.

20. Partner Offset-If a paired read’s partner hits to the same chromosome (as it will in the vast majority of cases) and contig (if the chromosome is split into contigs) then this number added to Match Position gives the alignment position of the read’s partner.

21. Partner Strand-To which strand did the paired read’s partner hit (“F” for forward or “R” for reverse, blank if no match).

22. Filtering-Did the read pass quality filtering? “Y” for yes, “N” for no.
Section Review

Pipeline Output
High level lane summary statistics:
  • Summary Table (Summary.htm)
    • Cluster Density
  • IVC Plots
    • Visualization of average base intensity across time
  • Percentage Passed Filters
    • Percentage of reads passing CHASTITY filter
  • Percentage Aligned and Average Alignment Score
    • Percentage of filtered reads that align
    • Dependent on read length and reference genome quality
  • Error Rate
    • Error estimation for reads passing filters
  • Errors.htm
    • Visualization of errors across run cycles

Output Files
• Intermediate output files
  • .int – raw un-normalized intensity, per tile
  • .seq – raw sequence calls, per tile

Final Output files
• *_realign.txt
  • Standard output from ELAND and PhageAlign
• *_export.txt
  • Standard output for Eland_extended
  • Contains all unfiltered reads in a lane
Hands-on Exercises
Introduction

In this Chapter of the Analysis Pipeline training, sample data sets are provided to allow the user to practice hands-on exercises in the Pipeline software suite.

Exercise #1—Running the Pipeline

This exercise explores the standard method of invoking the Analysis Pipeline on a data set. The basic steps are:

- Setup the configuration file,
- Run the goat_pipeline.py program to generate the makefiles, and
- Execute the make.

Steps:

1. Use cd to change directory to the root of the analysis folder.

   cd [DATA]/Sample_Data

2. In a UNIX text editor, create a configuration file for GERALD. Name the file config1.txt, and have it read as follows:

   ELAND_GENOME [GENOME]
   ANALYSIS eland
   USE_BASES all
   GENOME_FILE e coli.fa
   SEQUENCE_FORMAT --fasta

3. Test the makefile generation.

   [PIPELINE]/Goat/goat_pipeline.py Images --GERALD=config1.txt

If the test completes successfully, output similar to Figure 26 appears:
Figure 26: makefile Generation Output Example
Steps Cont.:

4. You can generate the makefiles by using the “--make” option. Generate makefiles.

[PIPELINE]/Goat/goat_pipeline.py Images --GERALD=config1.txt --make

5. Change directory to the newly created Firecrest folder.

cd Data/C1-37_Firecrest1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]

6. Execute the makefiles.

make -j[N] recursive

Entering this command will cause the pipeline software to begin running. The “-j[N]” part of the command tells the Analysis Pipeline how many processors are available to use, while the “recursive” part tells the Analysis Pipeline to continue going down the directory structure until the analysis is complete.

When you issue this command, you will see a great deal of text running across the screen. This screen output details what the Pipeline programs are doing. It is occasionally useful to examine this data. We will learn in a later step how to redirect this output to a file. For now, wait until the text has finished scrolling and the command prompt returns. This will be your indication that your first analysis run is finished.

7. Examine the output. The interesting output is found in the Gerald Directory. From the Firecrest directory:

    cd Bustard1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]
    cd Gerald1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]
    ls
Steps Cont.:

There are two types of output files. The HTML files contain summary run statistics, while the text files contain the final output. Use a web browser to open the summary.htm file (it is often simpler to have the UNIX data directory cross-mounted to your Windows PC to examine the HTML output). This file gives information about how the run went. Note how the summary lane statistics reflects a higher level of the data that is included in the expanded lane statistics table, which in turn contains a roll-up of information that is found in the detailed tile by tile tables.

The summary.htm also contains links to other important statistics files. Follow the IVC.htm link. This page shows thumbnails of the Intensity vs. Cycle graphs. These graphs give a quick overview of how your experiment performed. Each thumbnail image is a link that brings up a larger version of the image. Use the back button on your browser to return to the summary page, and then follow the link to the errors.htm page. On this page, thumbnails show the error rates associated with each tile in a lane. Again, these thumbnails link larger images.

In addition to the HTML summary pages, the Gerald directory contains many text files. These files contain the output from the alignment step. The files that will be of the greatest interest are the sequence files, the quality report files and the alignment files.

For each lane, a file called s_[LANE]_sequence.txt is created. This file contains the list of sequences in the lane which passed the quality filtering step. The format of this file is controlled by the “SEQUENCE_FORMAT” option in the Gerald configuration file. Since we specified “--fasta” we will find this file contains a FASTA library.

8. To confirm, in the UNIX window type:

    more s_1_sequence.txt

A single page of the file is output to the screen. The file is in FASTA format, with the name of the sequence as the header and the sequence following. The sequence name is composed of the run number concatenated with the lane number, tile number, and the X/Y coordinates of the cluster. Page through some of the output using the space bar, then type “q” to quit.
Steps Cont.:

If we had not specified the FASTA format, then the output would have included the base-by-base quality scores. In the default output mode, there are four lines for each sequence:

@70720_HWI-EAS00_8_1_138_896
CATGATACGGCGACC
+70720_HWI-EAS00_8_1_138_896
IIIIIIIIIIIIIIHH

In this example, the first and third (and all odd numbered) lines are the sequence headers. Header names prefixed with an "@" symbol are followed by the DNA sequence, while header names prefixed with the "+" symbol are followed by the coded quality score. To allow easy alignments, the default quality output is an ASCII symbol. Conversion back to a numeric score is accomplished simply by subtracting 64 from the ASCII code value. Thus the numeric score for the first base in the example is 41 (105 - 64).

Another useful file to look at is the quality report. This file is named s_[LANE]_qreport.txt. This report shows the distribution of the quality scores within the lane, either as a running total or as a discrete value. Run one of the qreport files through the “more” UNIX command to get feel for what this file looks like.

Finally, on a tile by tile basis view the alignment output for all the clusters. The alignment files are named s_[LANE]_[TILE]_align.txt. Use the more command to examine one of these files. The output will look something like Figure 27:

CAATGACAGCGGAACAATAGAGATCTGAAACAGCC 0 0
CTGATTTGTTTTTTCACGGTGTTGGCGATCTGGAACA 0 0
CGTTCATCAAAGGTCAGTCGCCCAGTCTGGCGATAG 18500 1 e_coli:1864461 R CTATCGCCAGACTGGGAGACTGACCTGGACCTTTGATGAACG 15359
CGCAGGTAAACGCAGTCAGCTTACGGGTATCGATATCG 17453 1 e_coli:31508 R CGATATCGATACCCGAAGCTGACGCGTTACTGCGT 15359
CGCCTGTTGCTTCCAGCTGGCTGGCTTGGCGCGT 17453 3

Figure 27: Output Example

This is a space-delineated file format. The first column is the sequence that was searched, the second is the best score found, and the third is the number of matches with that score. When there is one unique match, information about the match is given in the next three entries. These are the file and position, the orientation, and the matching sequence. The final column indicates the score of the next best match.
Exercise #2—Running a Second Alignment

On occasion the need run an alignment analysis over again. If you had to run the entire pipeline from scratch, this would become onerous. Fortunately you can run sections of the pipeline, reusing the data from earlier runs. In this exercise, you will learn how to run only part of the pipeline to change the analysis.

Steps:

1. Use cd to change directory to the root of the analysis folder.

   cd [DATA]/Sample_Data


   cp config1.txt config2.txt

3. In a UNIX text editor edit config2.txt to use an eland_extended analysis. The are two changes in the file. We have altered the "ANALYSIS eland" to "ANALYSIS eland_extended" and we have added a new parameter called EXPT_DIR, which contains the full path to the Bustard directory (note that page width limitations forced word wrap in the path—in the configuration file the path name should be on a single line). The file should read as follows:

   EXPT_DIR [DATA]/SAMPLE_DATA/Data/C1-37_Firecrest1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]/Bustard1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]ELAND_GENOME [GENOME]
   ANALYSIS eland_extended
   USE_BASES all
   GENOME_FILE e_coli.fa
   SEQUENCE_FORMAT --fasta

   **Note the first line in the above command is intended to be entered as a single line. Due to page size limitations it appears wrapped in the Student Guide.**
Steps Cont.:

4. Generate makefiles.

[PIPELINE]/Gerald/GERALD.pl config2.txt --FORCE

The GERALD.pl script is the portion of the GOAT pipeline which creates the Gerald directory and makefile. The new Gerald directory will be created under the Bustard directory specified by the EXPT_DIR parameter, and the makefiles will refer to the files in the same directory. Also, notice that we use the --FORCE option instead of --make. GERALD.pl uses --FORCE instead of --make to generate the make file.

5. Change the directory to the newly created GERALD folder.

    cd Data/C1-37_Firecrest1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]/Bustard1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]/Ger-
    ald1.8.28_[DAY]-[MONTH]-[YEAR]_[USER]

**Note the above command is intended to be entered as a single line. Due to page size limitations it appears wrapped in the Student Guide.

Note there will be two GERALD directories—the one created during exercise 1 and the new one. The two directories will be differentiated by the “[DAY]-[MONTH]-[YEAR]” section of the file name. If the date is the same, then the folders will be versioned by appending “.2” to the user name.

6. Execute the makefiles.

    make -j[N] . output.log

    Notice the difference between this make command and the make command shown in Exercise 1. We have appended “> output.log” to the command. In UNIX, this phrase is known as a “redirect” because it redirects the output from the screen into a file. Thus when you launched the command the cursor moved down under the command line and sat there blinking.

    The magic redirect character is the “>” and the file name can be anything. In general, the file is referred to as the “make log” or “log file.” If something goes wrong with the analysis, Illumina’s Technical Support will often ask you to send this file, so it is generally a good idea to generate the file and save it until you know the analysis was successful.

    When the analysis is finished, the command prompt will reappear.
Steps Cont:

7. Examine the output.

ls

The first thing that jumps out at you is that while the HTML files are unchanged, there is a completely different set of text output files. The simplified output is a strong reason to use the new eland_extended mode.

The final text output is now a file called s_[LANE]_export.txt, which contains the information formerly contained in the s_[LANE]_[TILE]_align.txt files, as well as information which was formerly available only in some of the intermediate files. It is also noteworthy that the export file contains all the sequences and alignments, regardless of quality; at the end of the line is a “Y” or “N” indicating if the sequence passed the filtering criteria.
Exercise #3—Skipping Cycles

Another common situation that arises is when the first few bases of a sequence are identical. For instance, this may occur when the DNA has been isolated with a set of adapters such that the first three bases sequenced are from the adapter. Two problems immediately arise. Because there is no variation in base composition, the cross talk matrix and phasing estimates will be skewed. Second, we know the adapter won’t align properly to the genomic sequence, so we will artificially increase our error rate. Thus, rather than attempting to use the adapter sequence in our analysis, we will instead drop the first three bases using the --cycles command to restrict the analysis to a specific set of cycles. We will be using ELAND, which we used in Exercise 1. For this reason we will be using config1.txt

Steps:

1. Use cd to change directory to the root of the analysis folder.

cd [DATA]/SAMPLE_DATA

2. Test the makefile generation.

[PIPELINE]/Goat/goat_pipeline.py --cycles=4-36 Images --GERALD=config1.txt

By default, all available cycles are used. Here, the system is restricted to start from cycle four, skipping the first three bases.

3. Generate makefiles.

[PIPELINE]/Goat/goat_pipeline.py --cycles=4-36 Images --GERALD=config1.txt --make

4. Change directory to the newly created Firecrest folder.

cd Data/C4-36_Firecrest1.8.28_[DAY]-[MONTH]-[YEAR]_Training

Note that the directory name reflects the abbreviated set of cycles. There is also no version number, as this is your first run using the abbreviated cycles.
5. **Execute makefiles.**

```
nohup make -j[N] recursive > pipeline.log
```

We have again added a new command to the make command: the `nohup` command. Normally when you log out, all the programs you are running are terminated. The `nohup` command tells the computer that the program should continue to run, even if you log out.

6. **Examine the output in summary.htm and one of the alignment files.**

Note there is little difference in the summary file. Again, this is because we are using a carefully controlled data set with good base composition throughout the sequence. Thus, our expectation is to have good quality sequence no matter where we start our cycles.

The sequence and alignment files are a slightly different story. Examine one of the sequence files. Note that the sequences are now shorter by three bases, just like we told the pipeline to do. Now examine one of the alignment files, preferably the analogous one to the ones you examined earlier in Exercise 1. Note the changes in both the scores and the alignment positions.
Appendix A—Installing the Data Set

The training data set is a compressed file on the distribution DVD.

Move and unpack the file sample_data.tar.gz into your Illumina data directory.

cd <Illumina_data_directory>
cp <path_to_DVD>/Sample_Data.tar.gz .
tar zxf Sample_Data.tar.gz
rm Sample_Data.tar

Next set up the genome library (after making sure the Sample_Data directory has been created and contains files).

cd <GenomeDir>
cp <path_to_DVD>/e_coli.fa.gz .
gunzip e_coli.fa.gz

Finally, set up the squashed library for ELAND

mkdir <ELAND_BASE>/sample
<PIPELINE>/Eland/squashGenome <ELAND_BASE>/sample
<GenomeDir>/e_coli.fa
## Appendix B—Helpful UNIX Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd dir</td>
<td>Change to directory dir</td>
</tr>
<tr>
<td>mkdir dir</td>
<td>Create new directory dir</td>
</tr>
<tr>
<td>rmdir dir</td>
<td>Remove directory dir</td>
</tr>
<tr>
<td>mv file dir</td>
<td>Move file to directory dir</td>
</tr>
<tr>
<td>mv file1 file2</td>
<td>Rename file1 as file2</td>
</tr>
<tr>
<td>ls</td>
<td>List files in directory</td>
</tr>
<tr>
<td>ls -lag</td>
<td>List files in detail</td>
</tr>
<tr>
<td>ps</td>
<td>Print process status statistics</td>
</tr>
<tr>
<td>pwd</td>
<td>Print working directory</td>
</tr>
<tr>
<td>nohup</td>
<td>Redirect output to file nohup.out</td>
</tr>
<tr>
<td>top</td>
<td>Show all active jobs</td>
</tr>
</tbody>
</table>

In addition, in most terminal programs, pressing the up arrow will bring back the last command you entered.
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAND</td>
<td>Acronym for Efficient Large-Scale Alignment of Nucleotide Databases.</td>
</tr>
<tr>
<td>Flow cell</td>
<td>A flow cell contains eight discrete lanes, each of which can handle a separate DNA sample. Prepared DNA samples are loaded into the flow cell and clonally amplified to clusters.</td>
</tr>
<tr>
<td>GERALD</td>
<td>Acronym for Generation of Recursive Analyses Linked by Dependency.</td>
</tr>
<tr>
<td>GOAT</td>
<td>Acronym for General Oligo Analysis Tool.</td>
</tr>
<tr>
<td>IPAR</td>
<td>Acronym for Integrated Primary Analysis and Reporting.</td>
</tr>
<tr>
<td>Lane</td>
<td>A lane is a discrete physical channel within a flow cell.</td>
</tr>
<tr>
<td>Phasing</td>
<td>In any given cycle, a few DNA molecules within a cluster may fail to extend. This is referred to as “phasing,” as the molecules which fail to extend are now out of phase with the rest of the cluster. Phasing can occur when a base is not incorporated, or a molecule does not deblock from the previous cycle.</td>
</tr>
<tr>
<td>Prephasing</td>
<td>A second type of chemistry error is known as “prephasing.” Prephasing occurs when an unblocked base is incorporated in the growing DNA molecule, allowing a second nucleotide to be incorporated. The resultant strand is then a base ahead.</td>
</tr>
<tr>
<td>Tile</td>
<td>During the sequencing cycles, the Genome Analyzer takes a series of images. Each imaged area is called a tile. Lanes are imaged in two or three columns of up to 110 tiles.</td>
</tr>
</tbody>
</table>