SciLifeLab

NBIS
NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN

PacBio Assembly

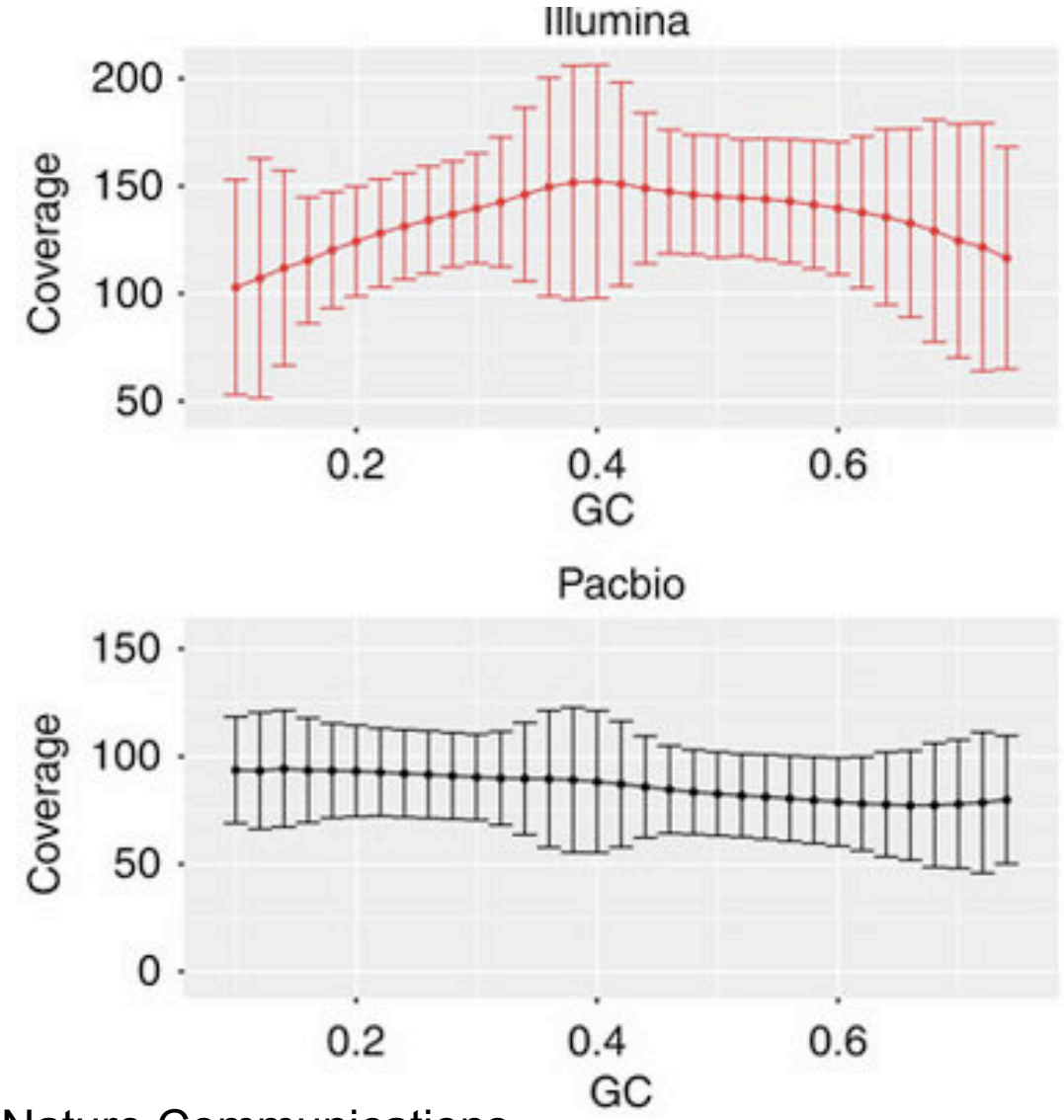
elixir excelerate

Outline

SciLifeLab

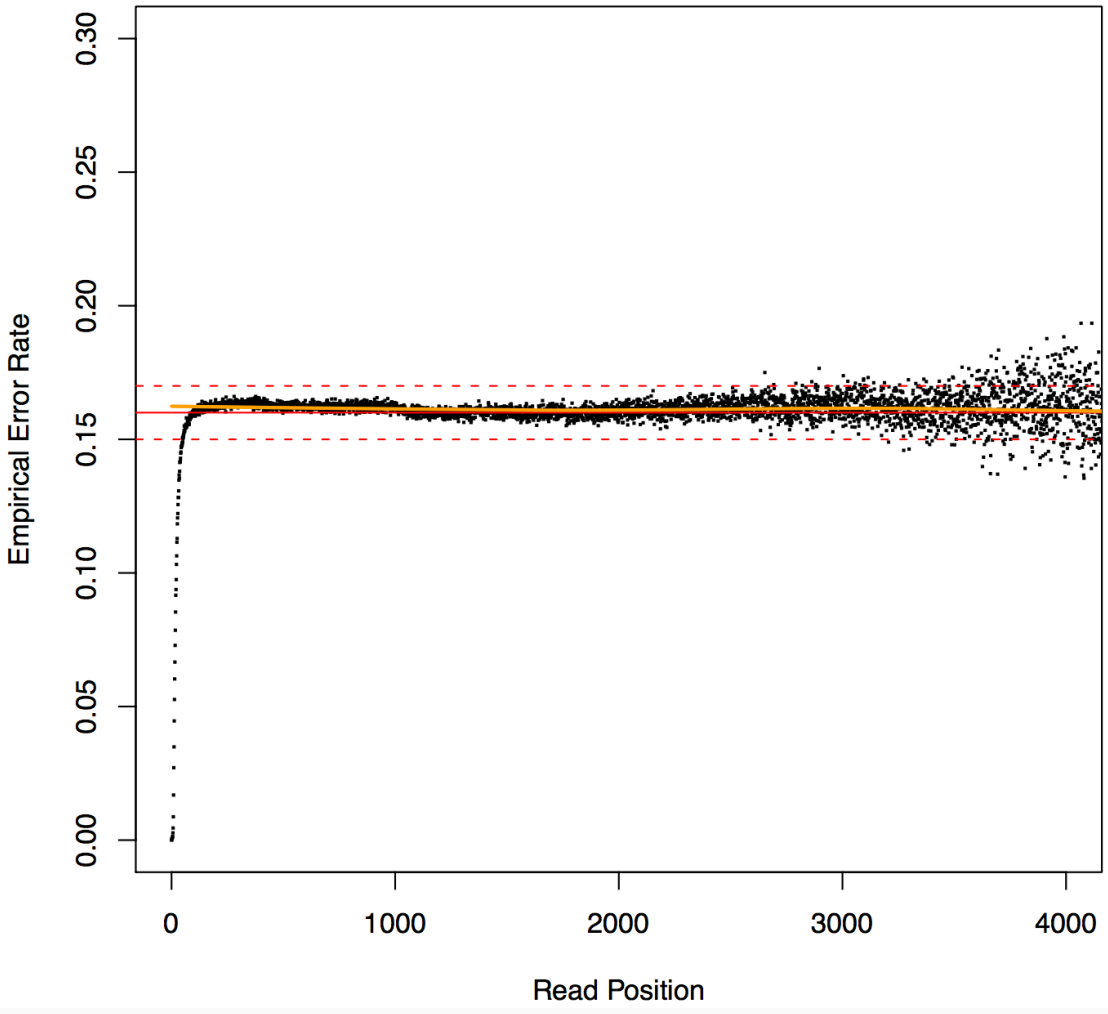
- The Error Profile of PacBio reads
- Methods of read correction
  - Correction with Illumina reads
  - Correction using PacBio reads
- Assembly Tools
- Assembly Diagnostics
- Assembly Polishing

PacBio GC Bias



Shi et al. 2016. Nature Communications.

PacBio Error Profile

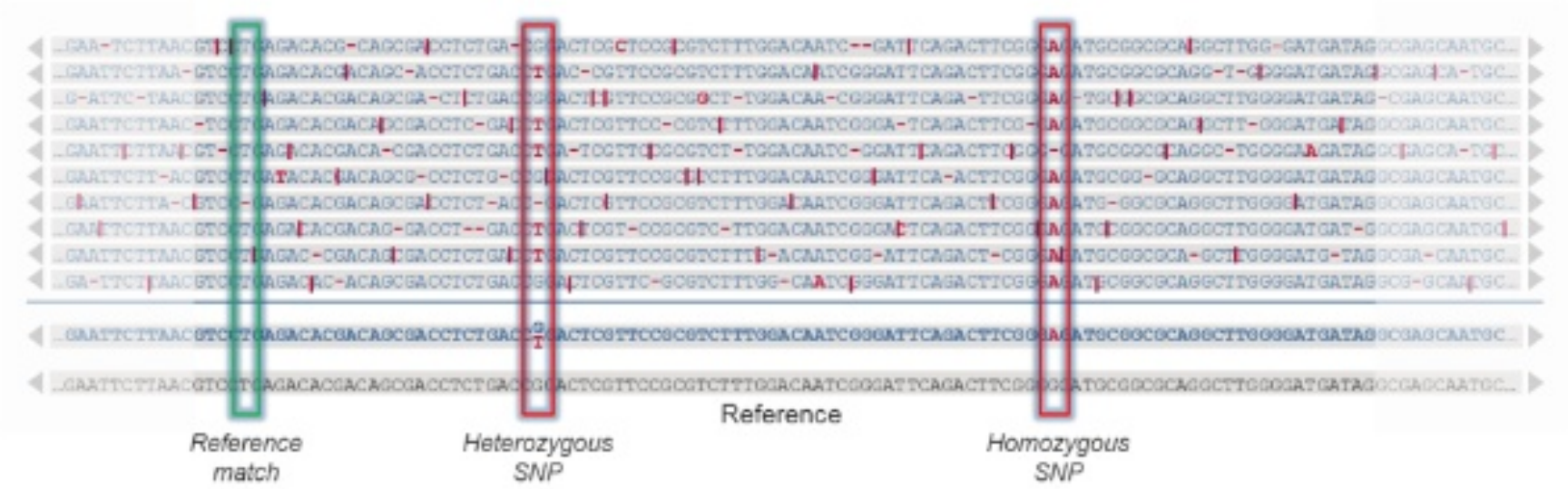


Koren et al. 2012. Nature Biotechnology.

PacBio Error Profile

SciLifeLab

PacBio: error rate



Single read: 86%

30x Consensus: 99.999%

T. Seemann. 2015: slideshare.net: Long read sequencing

PacBio Error Profile

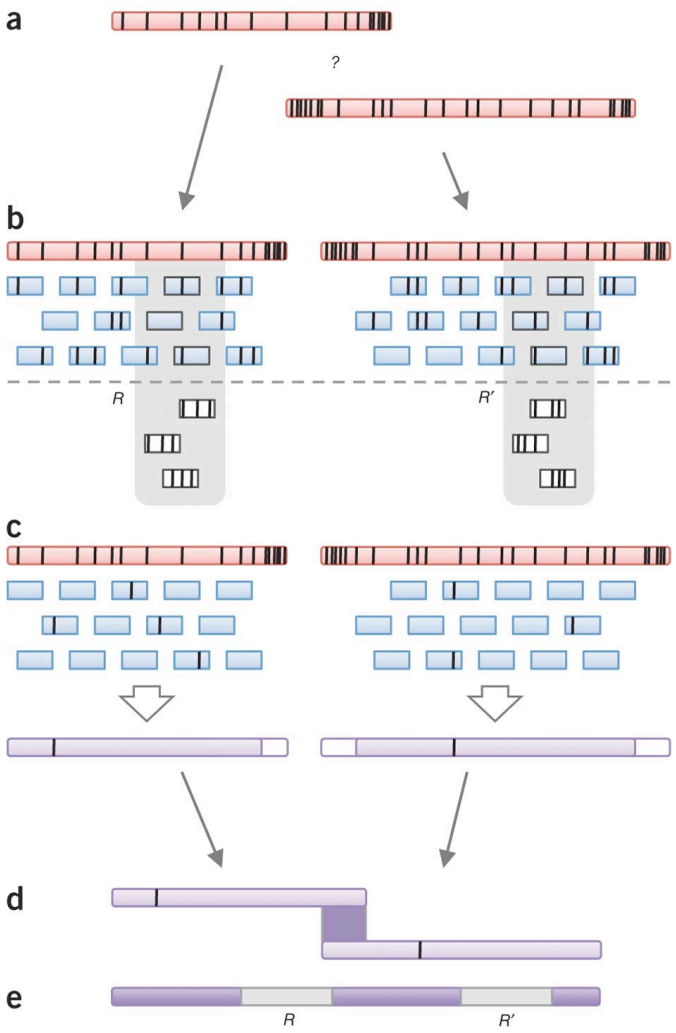
SciLifeLab

- Some statistics from sequencing the 16S rRNA gene.
  - Reads of Insert (>3 passes) - average sequence error rate of 0.65%
  - Insertions, deletions, and substitutions accounted for 31.2, 17.9, and 50.9% of those errors, respectively.
  - Substitution errors were equally likely
  - All four bases were equally likely to be insertion errors
  - G (39.4%) and A (24.3%) were more likely to be deleted than C (18.3%) or T (18.0%)
  - Percentage of base calls that had max quality did not vary among correct base calls (80.5%), substitutions (80.0%), or insertions (80.4%)
    - Quality values cannot be used to screen sequence quality

- Nearly random errors.

Schloss et al. 2016: PeerJ.

Read Correction



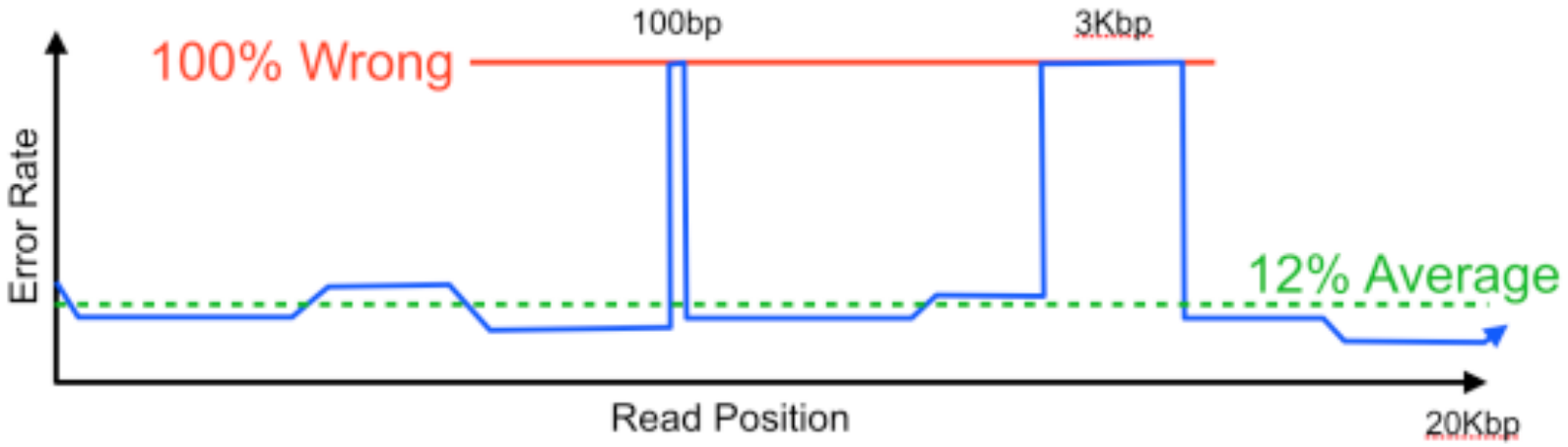
- Correction using Illumina reads
- Homopolymer correction, point mutations, and indels
- Doesn't correct structural errors
- Tools
  - PBcR / PacBioToCA
  - LSC / LSCplus
  - LoRDEC (de Bruijn graph)
  - Proovread
  - ECTools
  - Jabba (de Bruijn graph)

Koren et al. 2012. Nature Biotechnology.

Read Correction

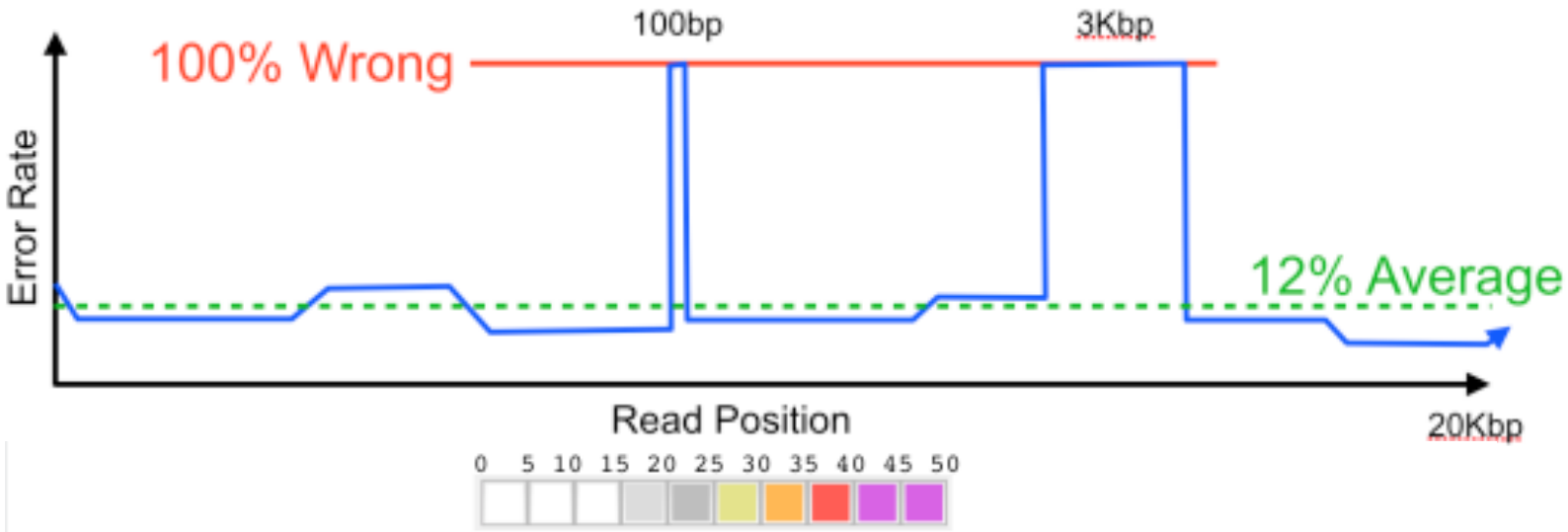
SciLifeLab

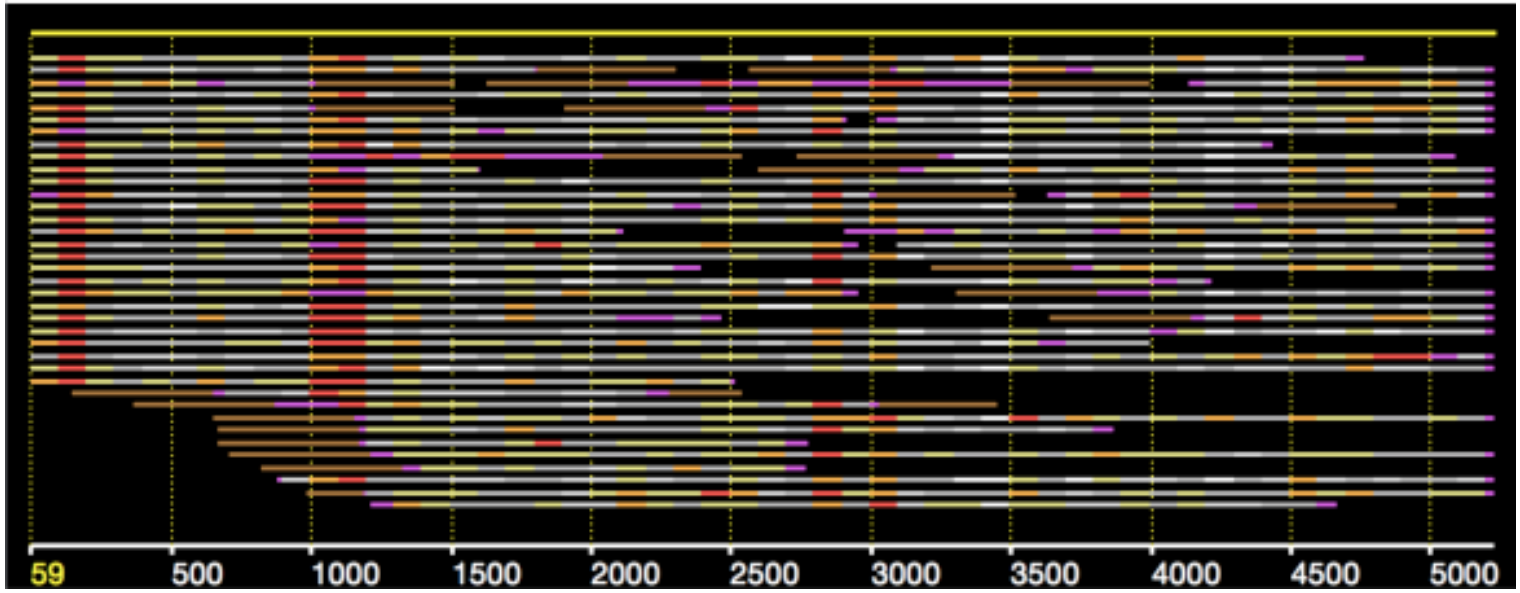
- Structural errors
  - Chimeric reads (see Tallon et al. 2014. BMC Genomics)
  - Missed or incorrectly inferred adapter
  - Interference from other molecules
  - …



G. Myers. 2015. Dazzlerblog.wordpress.com. Intrinsic Quality Values

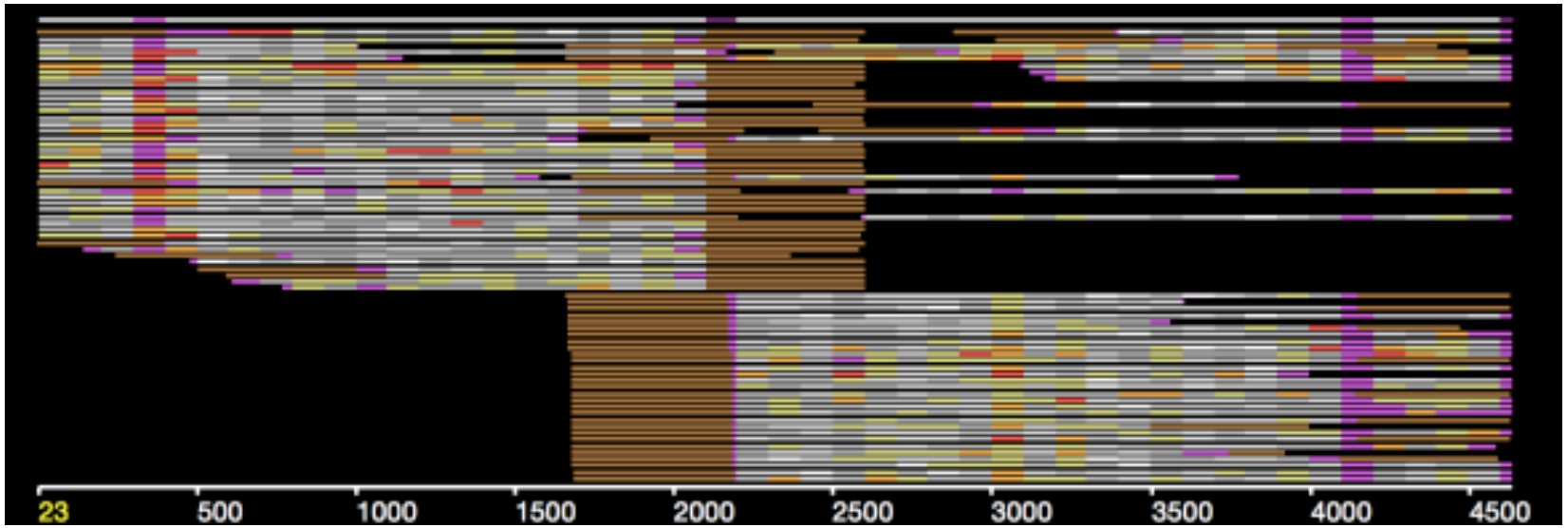
Read Correction

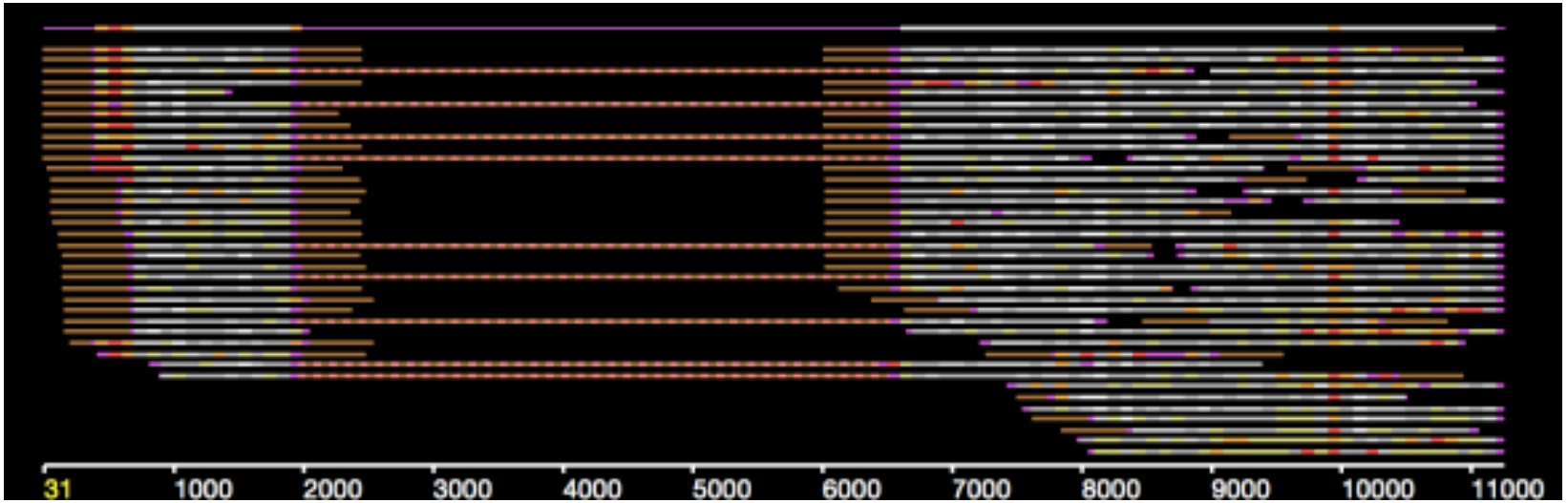




Read Correction

SciLifeLab

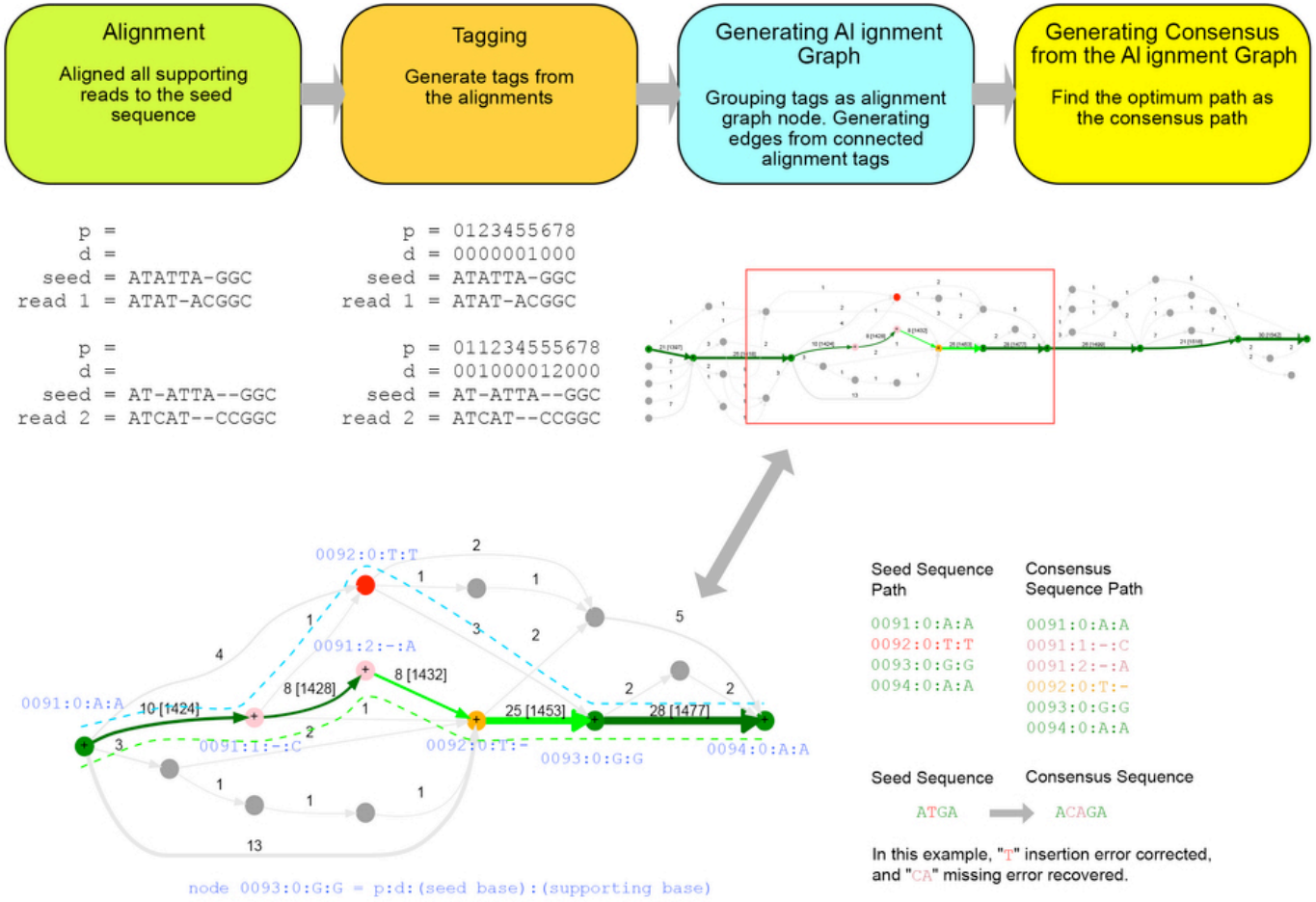




Read Correction

SciLifeLab

- Use a weighted directed acyclic graph to find consensus sequence



Supplementary Figure 12

An Example of how the FALCON-sense algorithm generates consensus sequence.

Chin et al. 2016. Nature Methods

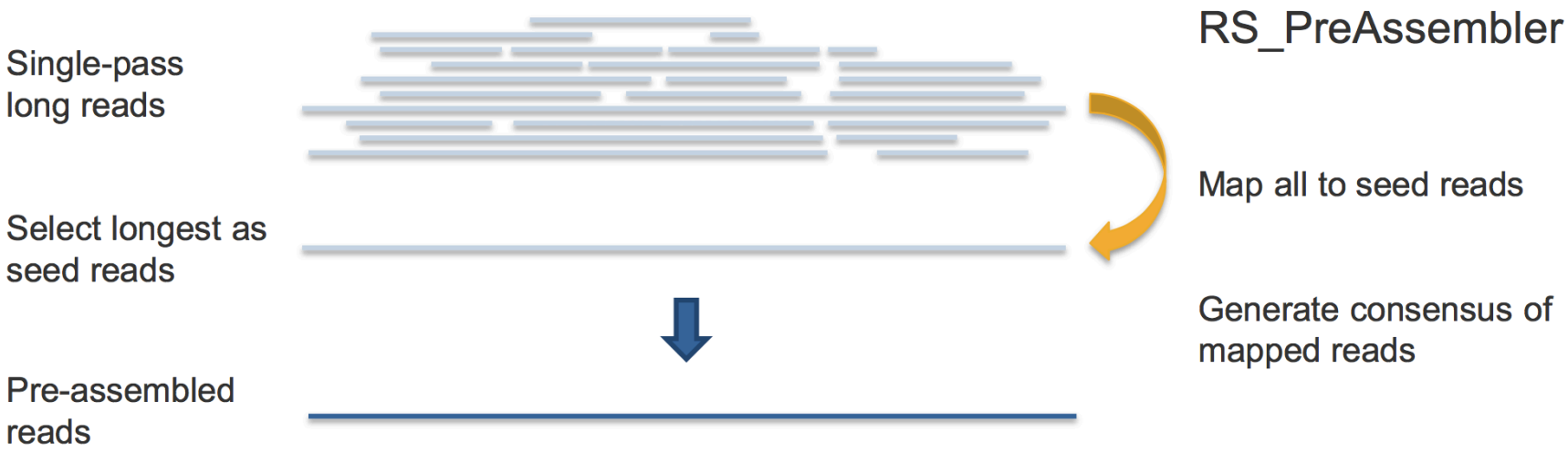
Read Assembly

SciLifeLab

- Popular PacBio assemblers:
  - HGAP
    - Limited to genomes < 200MB
    - http://www.pacb.com/support/software-downloads/
  - Canu
    - Large genomes
    - https://github.com/marbl/canu
  - Falcon
    - Large genomes
    - https://github.com/PacificBiosciences/FALCON-integrate
  - Miniasm
    - Large genomes
    - https://github.com/lh3/miniasm

HGAP

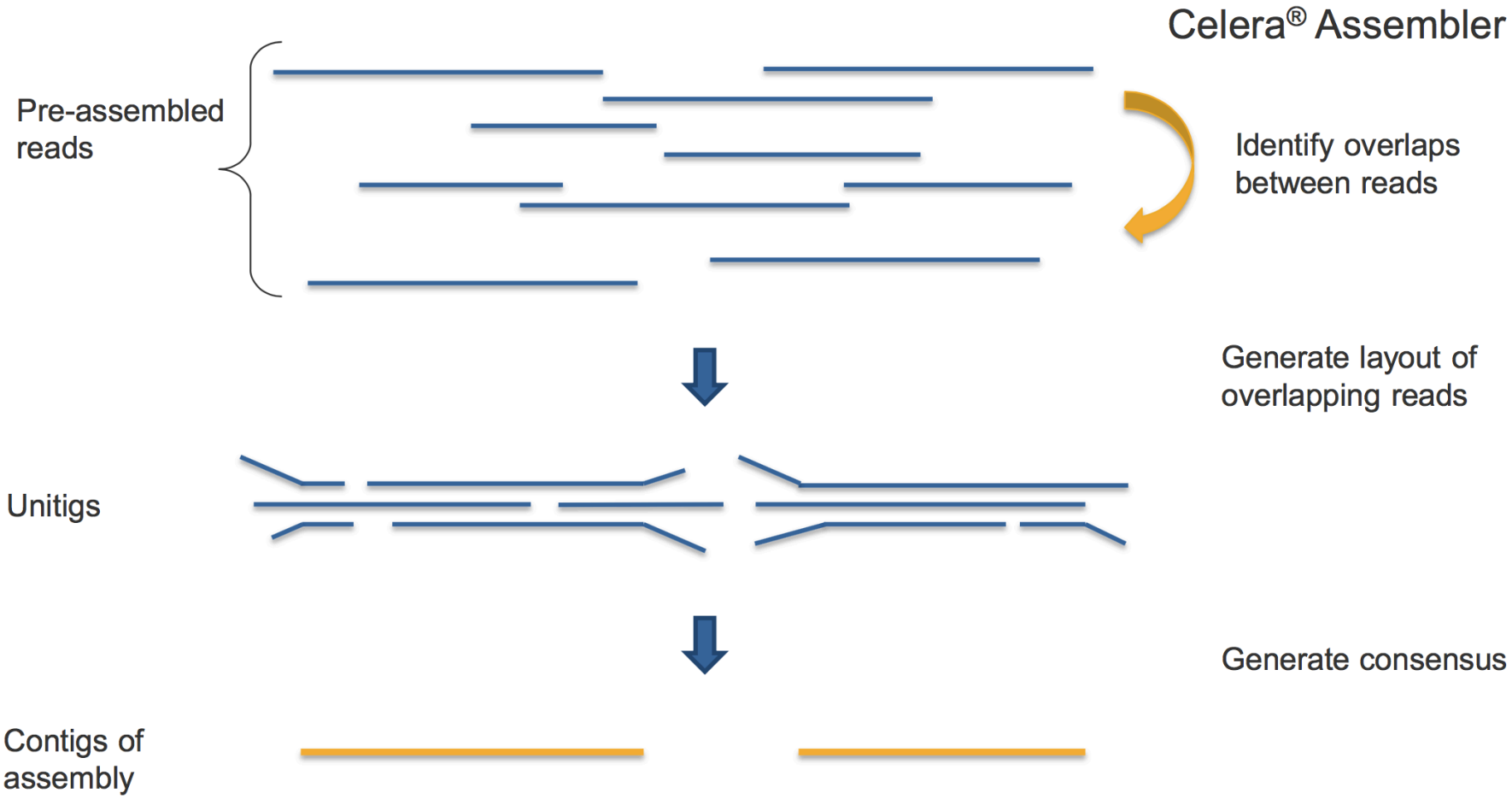
SciLifeLab



Slides from PacBio: BFX_Workshop_Oct_2014_Final-lh.pdf

HGAP

SciLifeLab



HGAP

SciLifeLab

Single-pass long reads

Contigs

High-quality consensus

RS_Resequencing

Map to *de novo*-assembled reference

Base-quality-aware consensus of uniquely mapped reads

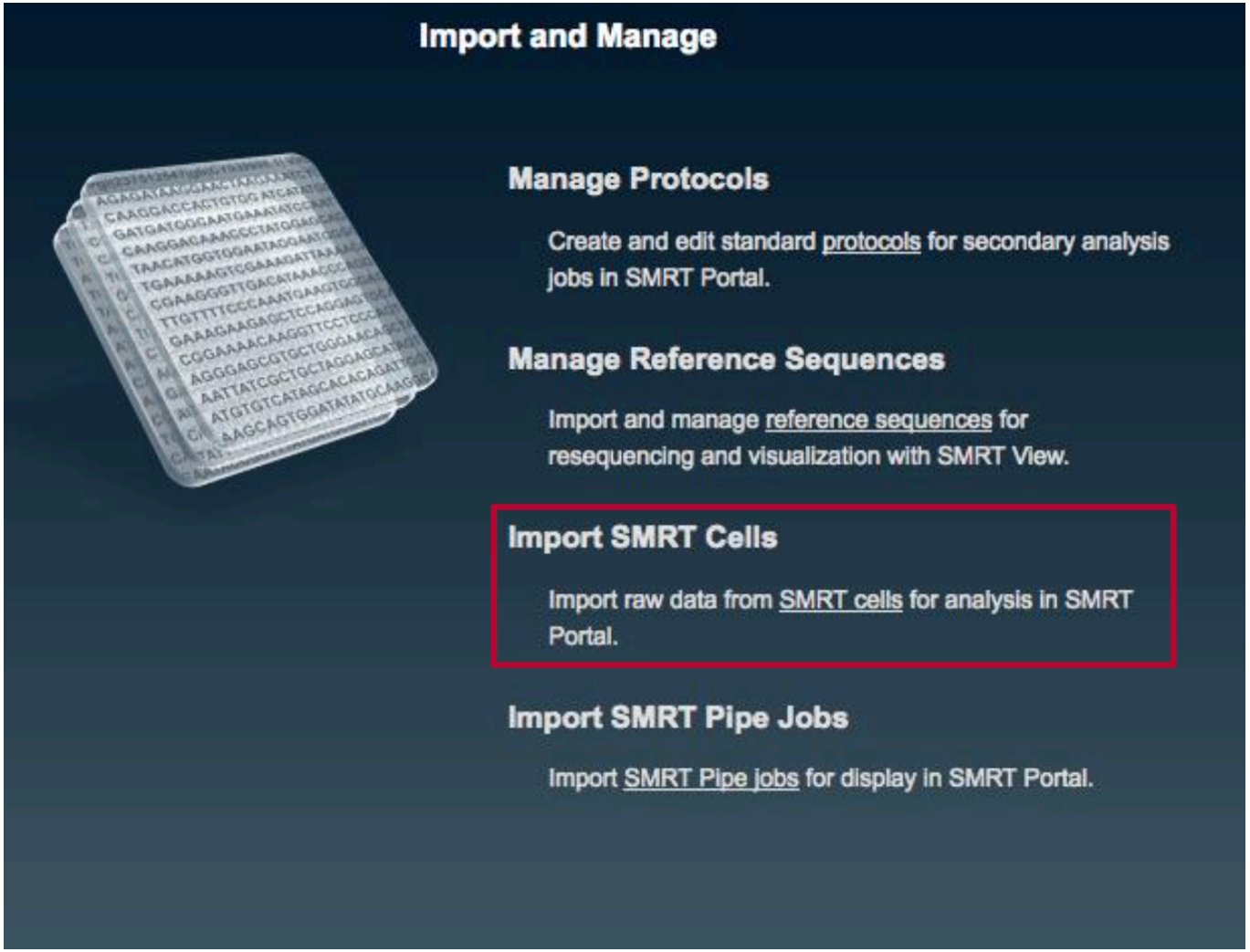
Quiver

HGAP

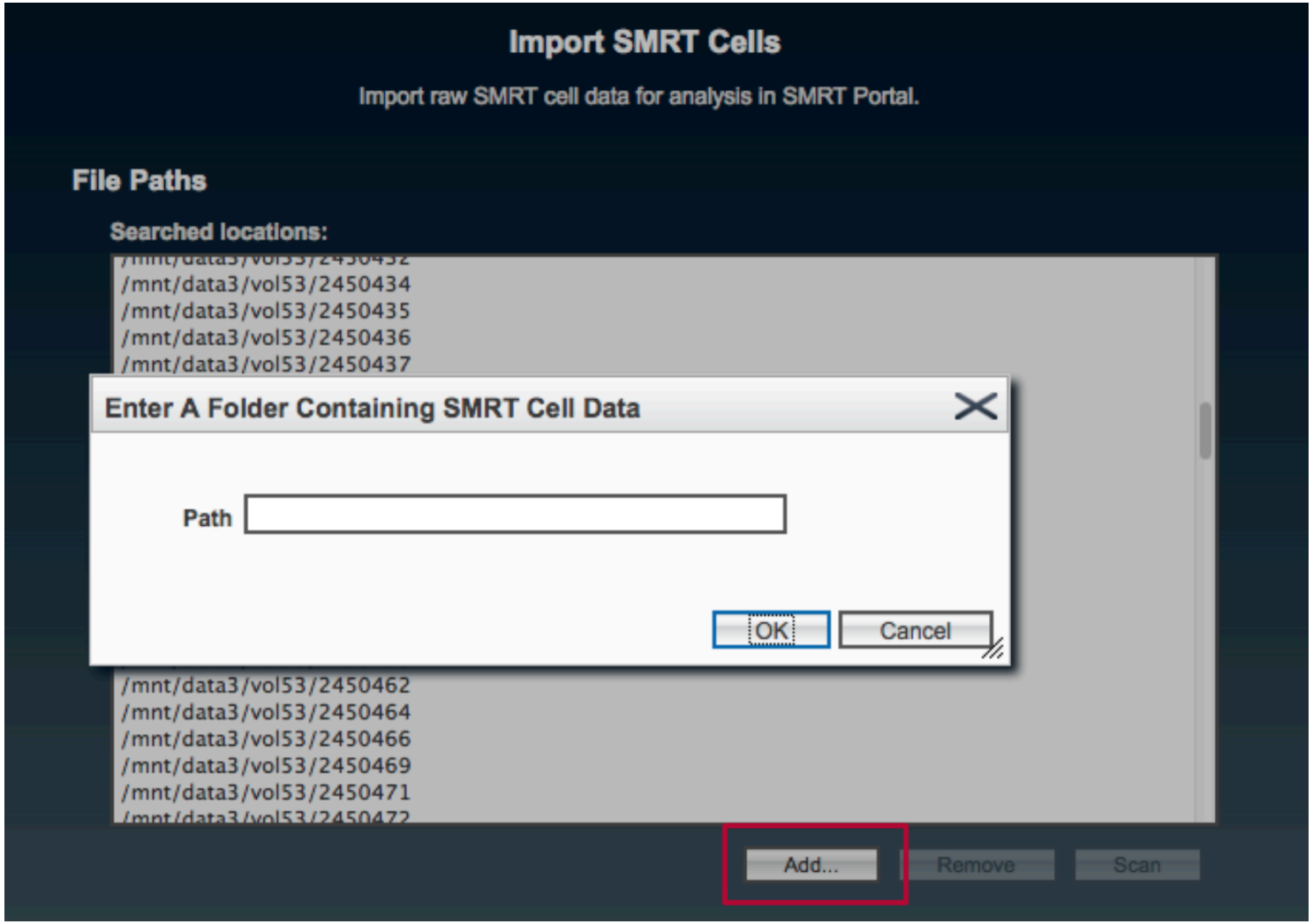
SciLifeLab

HGAP

SciLifeLab

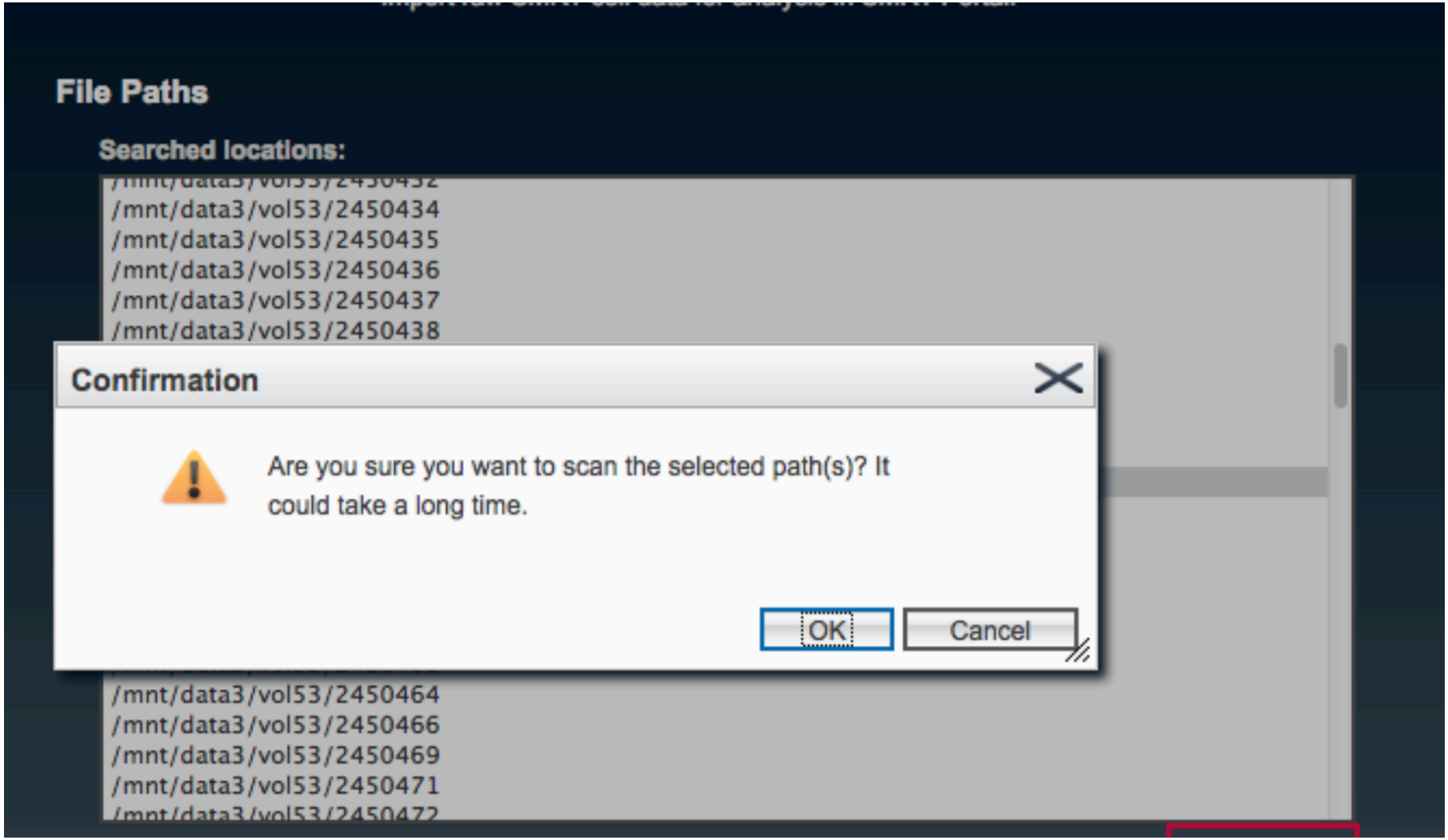


HGAP


Import SMRT Cells
Import raw SMRT cell data for analysis in SMRT Portal.
File Paths
Searched locations:
/mnt/data3/vol53/2450434
/mnt/data3/vol53/2450435
/mnt/data3/vol53/2450436
/mnt/data3/vol53/2450437
Enter A Folder Containing SMRT Cell Data
Path
OK
Cancel
/mnt/data3/vol53/2450462
/mnt/data3/vol53/2450464
/mnt/data3/vol53/2450466
/mnt/data3/vol53/2450469
/mnt/data3/vol53/2450471
Add...
Remove
Scan

HGAP

SciLifeLab

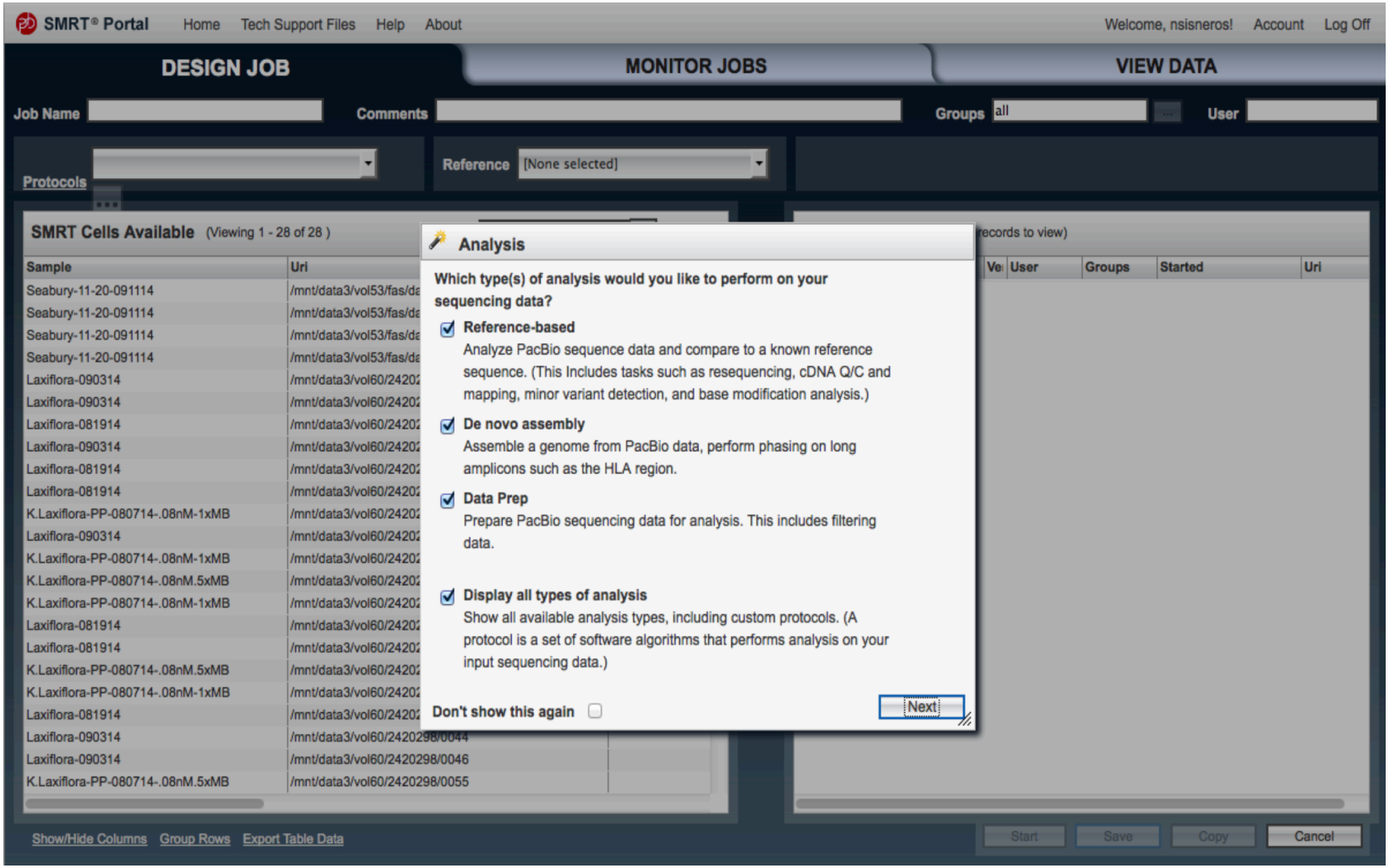


HGAP


SMRT® Portal
Home
Admin
Help
About
Welcome, administrator!
Account
Log Off
DESIGN JOB
MONITOR JOBS
VIEW DATA
Open Existing
Create New
Import and Manage
RECENT JOBS
Job Name
Protocol
Reference Sequence
Started
Status
User
ugmExampleA_reseq
RS_Resequencing.1
ugmExampleA
Completed
kluong
NHGRI10_4cells_tandemArt
HGAP_Assembly_Artifact.1
Completed
kluong

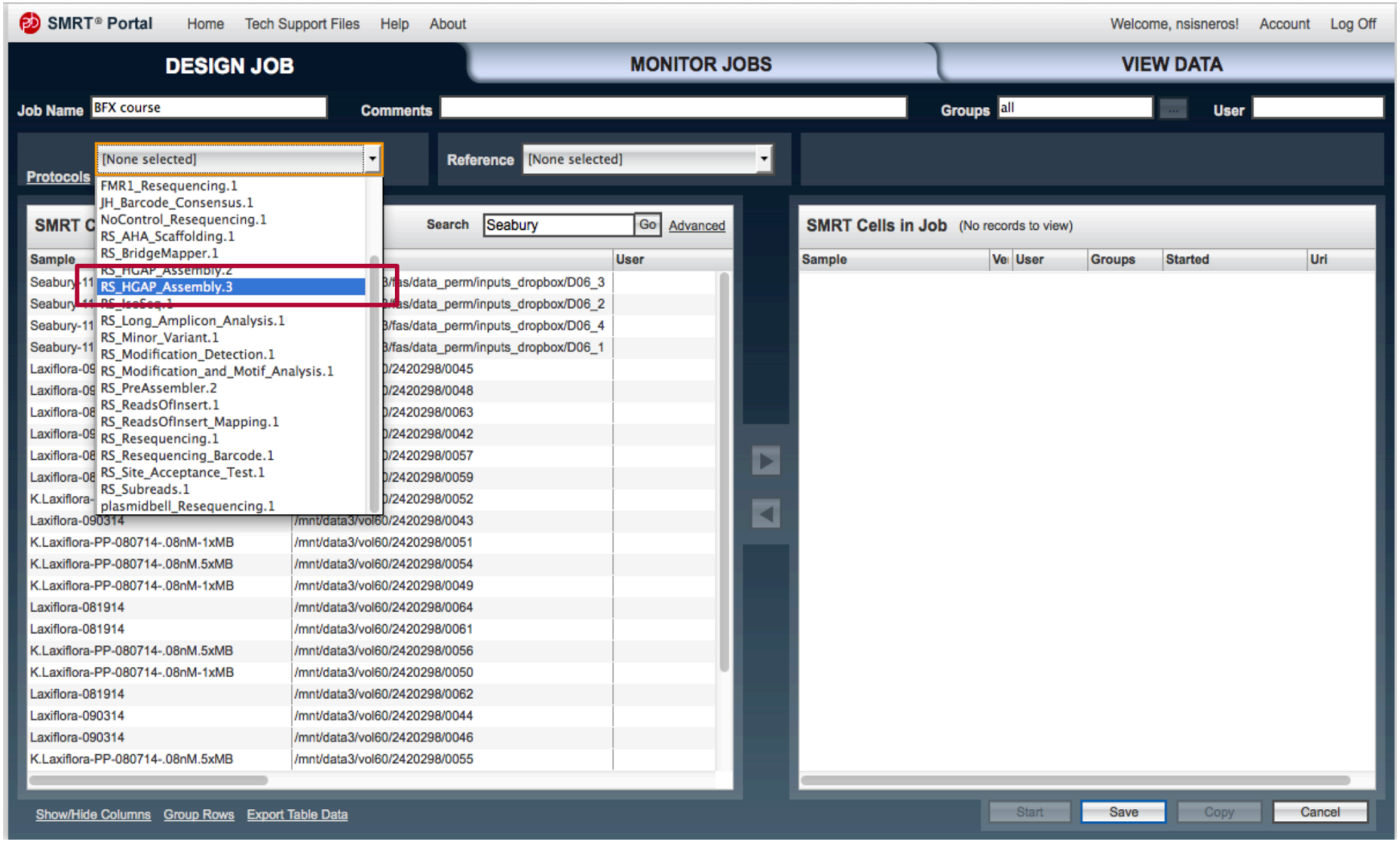
HGAP

SciLifeLab

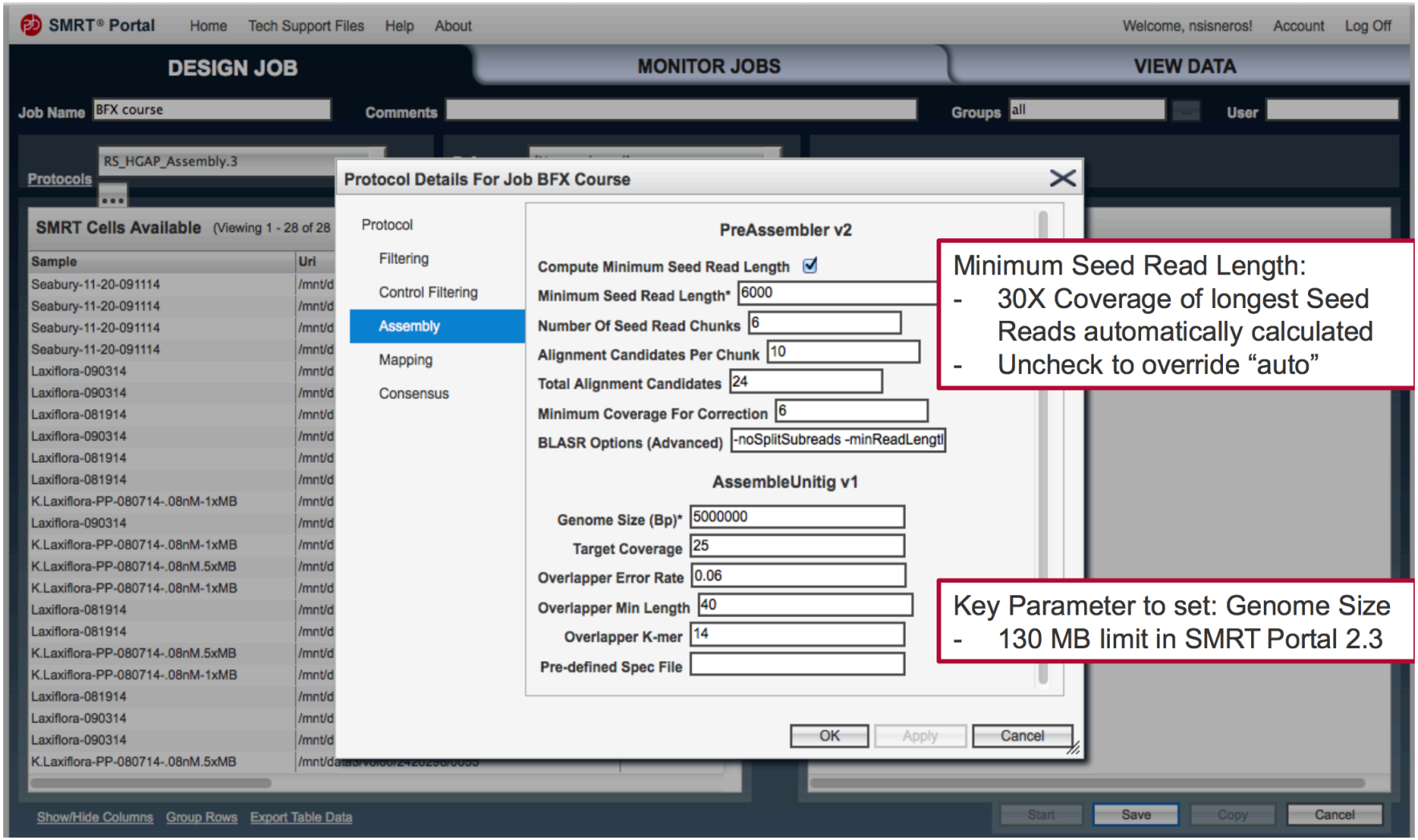


HGAP

SciLifeLab

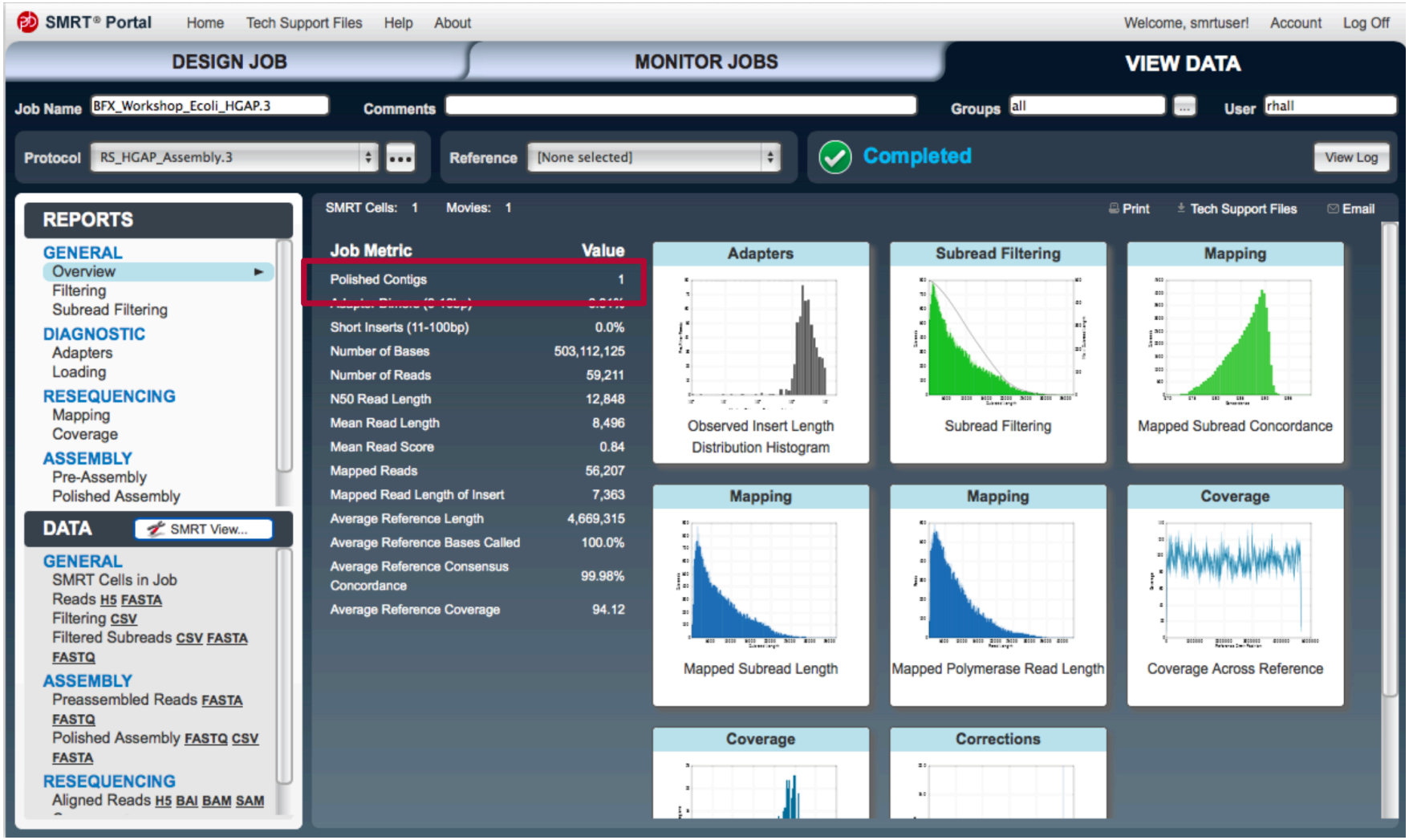


HGAP


SMRT® Portal
Home
Tech Support Files
Help
About
Welcome, nsisneros!
Account
Log Off
DESIGN JOB
MONITOR JOBS
VIEW DATA
Job Name
BFX course
Comments
Groups
all
User
Protocols
RS_HGAP_Assembly.3
SMRT Cells Available (Viewing 1 - 28 of 28
Sample
Uri
Protocol Details For Job BFX Course
Protocol
Filtering
Control Filtering
Assembly
Mapping
Consensus
PreAssembler v2
Compute Minimum Seed Read Length
Minimum Seed Read Length* 6000
Number Of Seed Read Chunks 6
Alignment Candidates Per Chunk 10
Total Alignment Candidates 24
Minimum Coverage For Correction 6
BLASR Options (Advanced) -noSplitSubreads -minReadLengtl
AssembleUnitig v1
Genome Size (Bp)* 5000000
Target Coverage 25
Overlapper Error Rate 0.06
Overlapper Min Length 40
Overlapper K-mer 14
Pre-defined Spec File
OK
Apply
Cancel
Minimum Seed Read Length:
- 30X Coverage of longest Seed Reads automatically calculated
- Uncheck to override “auto”
Key Parameter to set: Genome Size
- 130 MB limit in SMRT Portal 2.3
Show/Hide Columns
Group Rows
Export Table Data
Start
Save
Copy
Cancel

HGAP

SciLifeLab



HGAP

SciLifeLab

- Running HGAP (Command line)
  - Install SMRT Analysis software
  - Make a HGAP assembly job using the SMRT portal and save.
  - Save the settings.xml file as HGAP_protocol.xml

- Every SMRT Portal job has the following structure. **Example:**

```
/path/to/smrtanalysis/userdata/jobs/016/016234
├── data/
├── results/
├── log/
├── workflow/
├── job.sh
├── input.xml
└── settings.xml
```

  - `data` is a **directory** that contains intermediate and final data files for the analysis job
  - `results` is a **directory** that contains summary statistics and plots for the analysis job
  - `log` is a **directory** that contains all log files for the analysis job
  - `workflow` is a **directory** that contains all the executables for the analysis job
  - `job.sh` is an executable file used by SMRT Portal to run the `smrtpipe.py` analysis job
  - `input.xml` is a .xml file containing a list of input `bax.h5` files used to run the analysis job
  - `settings.xml` is a .xml file containing the parameters needed to perform the analysis job

HGAP

SciLifeLab

- Running HGAP (Command line) cont'd.
  - Modify Genome size in HGAP_protocol.xml
    - <param name="genomeSize" label="Genome Size (bp)">
      <value>5000000</value>
  - Source the SMRT analysis environment
    - source /path/to/smrtanalysis/install/smrtanalysis_2.3.0.140936/etc/setup.sh
  - Add the full paths of your raw data (*.bax.h5) into an input.fofn
    - find <data_dir> -name "*.bax.h5" > input.fofn
  - Convert the input.fofn to an input.xml
    - fofnToSmrtpipeInput.py input.fofn > input.xml
  - Run SMRT pipe using the protocol and input xmls.
    - smrtpipe.py --params=HGAP_protocol.xml xml:input.xml
  - Results are found in index.html in the working directory
  - Assembly is in data/polished_assembly.fastq.gz

HGAP

SciLifeLab

```
#! /bin/bash
#SBATCH -A <your uppmax project>
#SBATCH -p core
#SBATCH -n 8
#SBATCH -t 1-00:00:00
#SBATCH -J run_smrt_assembly
#SBATCH -e run_smrt_assembly-%j.out
#SBATCH -o run_smrt_assembly-%j.out

module load bioinfo-tools SMRT/2.3.0
WORK_DIR=$SNIC_TMP/smrt_assembly_$(date +%Y_%m_%d-%H.%M)
PROJ_DIR=$PWD
PROTOCOL_XML=$PROJ_DIR/Settings/HGAP_protocol.xml
DATA_DIR=${PROJ_DIR}/00_RawData         # Use full path
GENOME_SIZE=5000000

# Modify Protocol xml to the correct genome size
perl -0777 -i.original -pe "s/<param name=\"genomeSize\" label=\"Genome Size \(bp\)\">\n\s+<value>\d
+<\/value>/<param name=\"genomeSize\" label=\"Genome Size (bp)\">\n\t\t<value>$GENOME_SIZE<\/value>/
igs" $PROTOCOL_XML

# Activate SMRT Analysis environment
source $SMRT_SETUP_SCRIPT
mkdir -p $WORK_DIR; cd $WORK_DIR

# Make input file
find ${DATA_DIR} -name "*.bax.h5" > input.fofn
fofnToSmrtpipeInput.py input.fofn > input.xml

smrtpipe.py --params=$PROTOCOL_XML xml:input.xml

cd $PROJ_DIR; rsync -av $WORK_DIR .
```

Canu

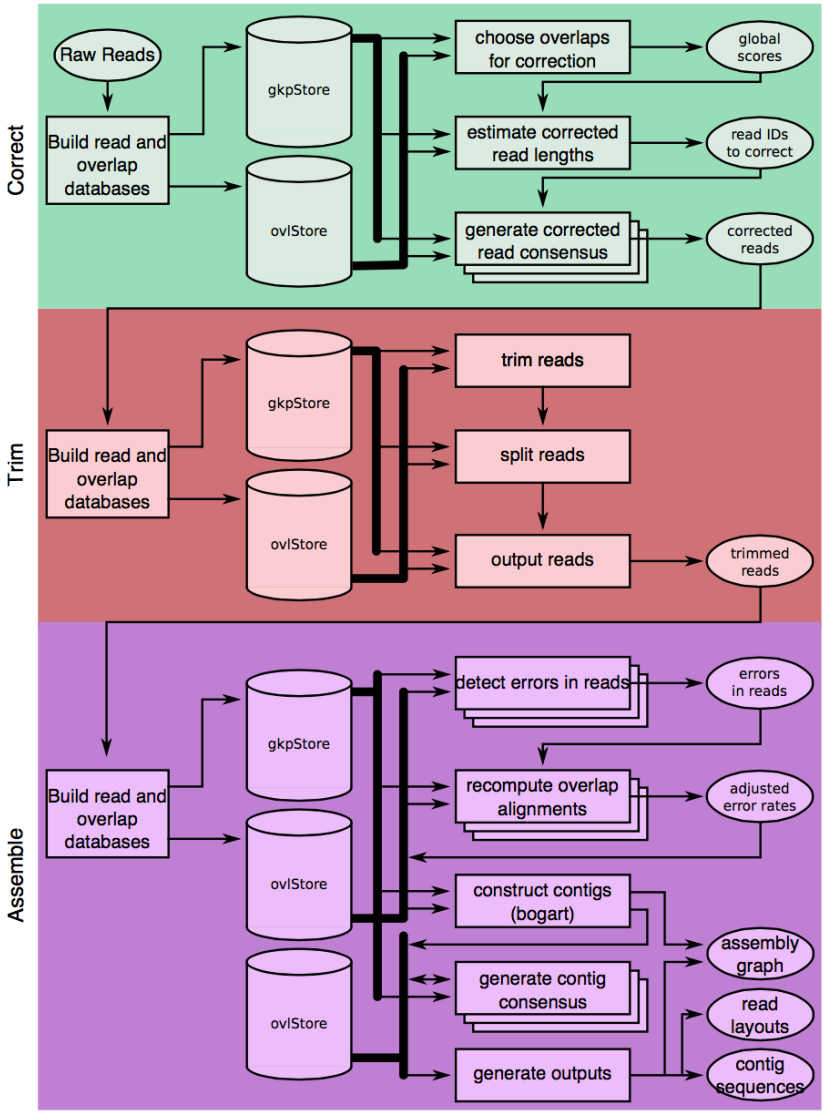


Figure 1. A full Canu run includes three stages: correction (green), trimming (red), and assembly (purple). Canu stages share an interface for binary on-disk stores (databases) as well as parallel store construction. In all stages, the first step constructs an indexed store of input sequences, generates a k-mer histogram, constructs an indexed store of all-vs-all overlaps, and collates summary statistics. The correction stage (green) selects the best overlaps to use for correction, estimates corrected read lengths, and generates corrected reads. The trimming stage (red) identifies unsupported regions in the input and trims or splits reads to their longest supported range. The assembly stage (purple) makes a final pass to identify sequencing errors; constructs the best overlap graph; and outputs contigs, an assembly graph, and summary statistics.

Koren et al. 2016. BioRxiv

Canu

SciLifeLab

- Running Canu
  - Can autodetect cluster settings (not recommended for milou)
  - Run canu on a node
    - useGrid=false
    - maxThreads=$NPROCS

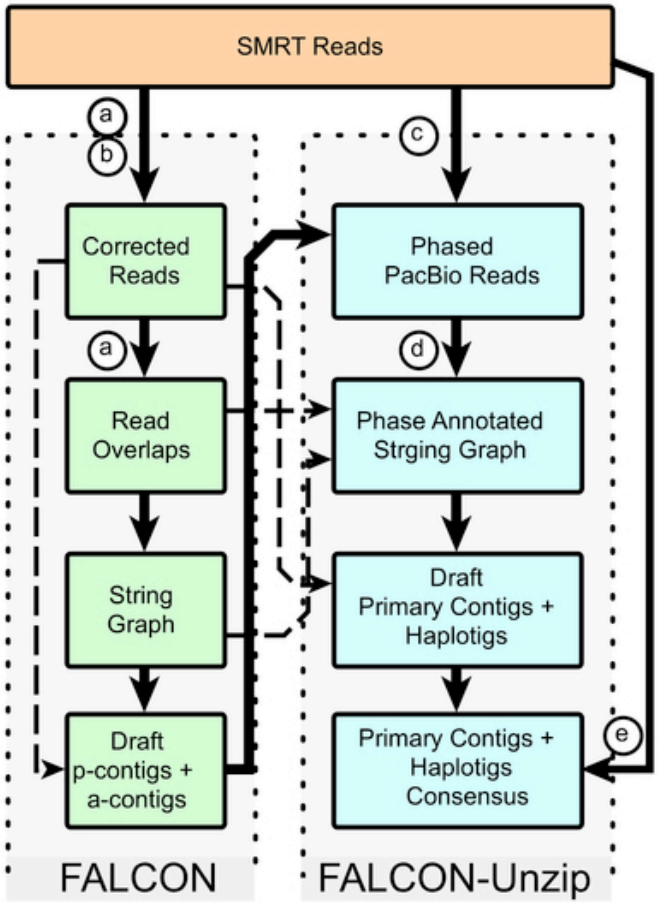
```
canu -p <file_prefix> -d <out_dir> genomeSize="18m"
   maxThreads=24 useGrid=false -pacbio-raw
   <filtered_subreads.fastq.gz>
```

- Results
  - Sequence is in file_prefix.contigs.fasta
  - Assembly graph is in file_prefix.gfa

Falcon

SciLifeLab

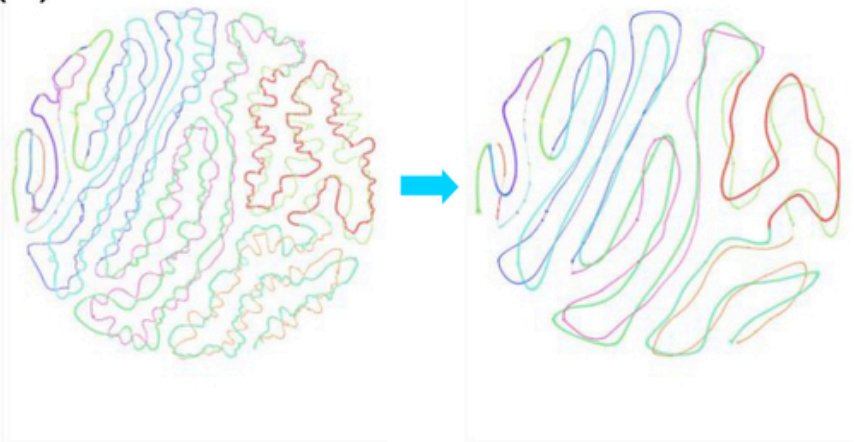
(a)



External code and internal modules used in FALCON and FALCON-Unzip

- (a) Daligner
- (b) Consensus Module (FALCON-sense)
- (c) Phasing Module (FALCON-phasing)
- (d) Graph "Unzip" Module
- (e) BLASR Alignment+ Quiver Consensus Module

(b)



Chin et al. 2016. Nature Methods

Falcon

SciLifeLab

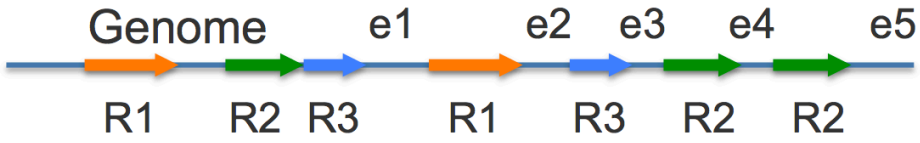
BIOINFORMATICS

Vol. 21 Suppl. 2 2005, pages ii79–ii85
doi:10.1093/bioinformatics/bti1114

Genes and Genomes

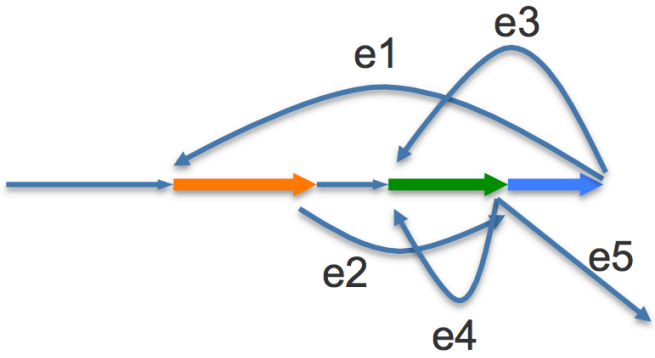
The fragment assembly string graph

Eugene W. Myers
Department of Computer Science, University of California, Berkeley, CA, USA



- String graph:
  - A graph structure that models a genome
- Nodes:
  - Particular positions (typically corresponding to the beginnings or endings of the read fragments) in the genome
- Edges:
  - The sequence between the vertices
- Any string from a path spell out a possible assembly from the reads

String Graph



Slides from Jason Chin:Diploid_Assembly_V1106_SpeakerDeck.pdf

Falcon

SciLifeLab

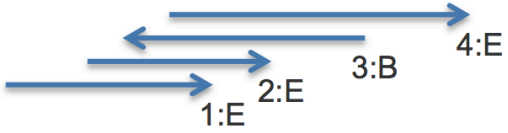
For each overlap, two edges are constructed.

Example:

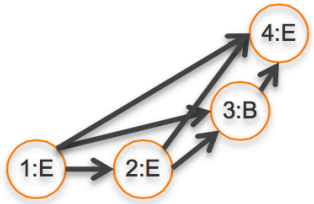
Overlapped reads

New edges

Add f:B, g:B, f:E, g:E as vertices
Add edges f:E ➔ g:B and g:E ➔ f:B

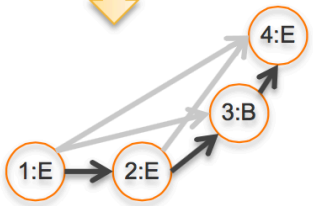


overlaps



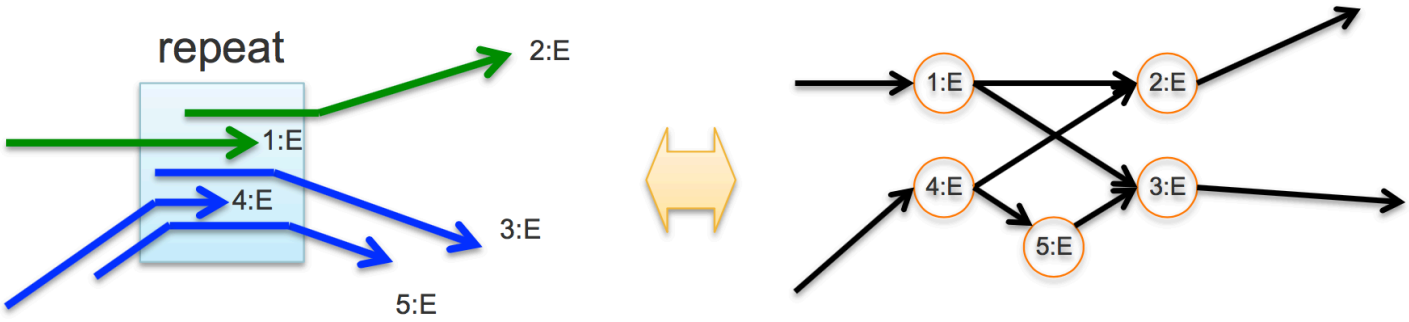
Initial graph

Transitive Reduction



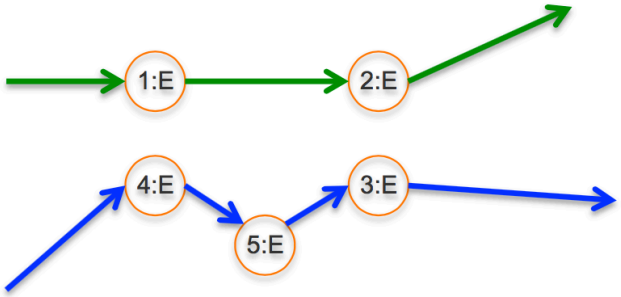
String graph

Falcon

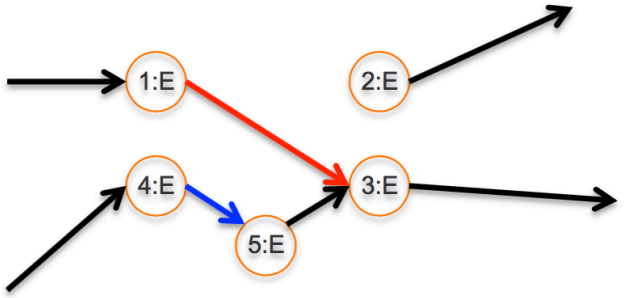


- Using a simple “best overlapping logic” to “untangle” the knots.

Desired final graph



Best overlap string graph



The 4E➔5E edge is better than 4E ➔2E.
The 1E➔3E edge is better than 1E ➔2E.
(“wrong” edge)

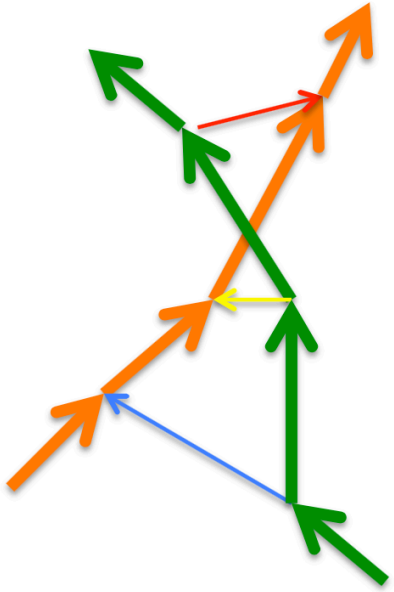
Falcon

SciLifeLab

String Graph

Unitig Graph

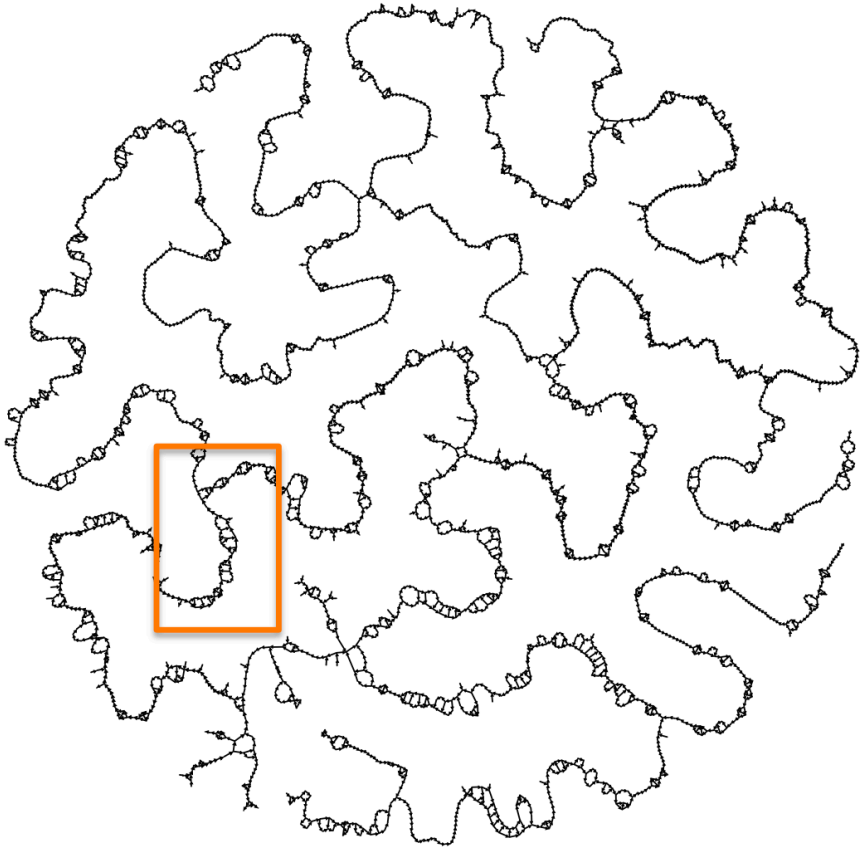
Graph traversal
for generating contigs



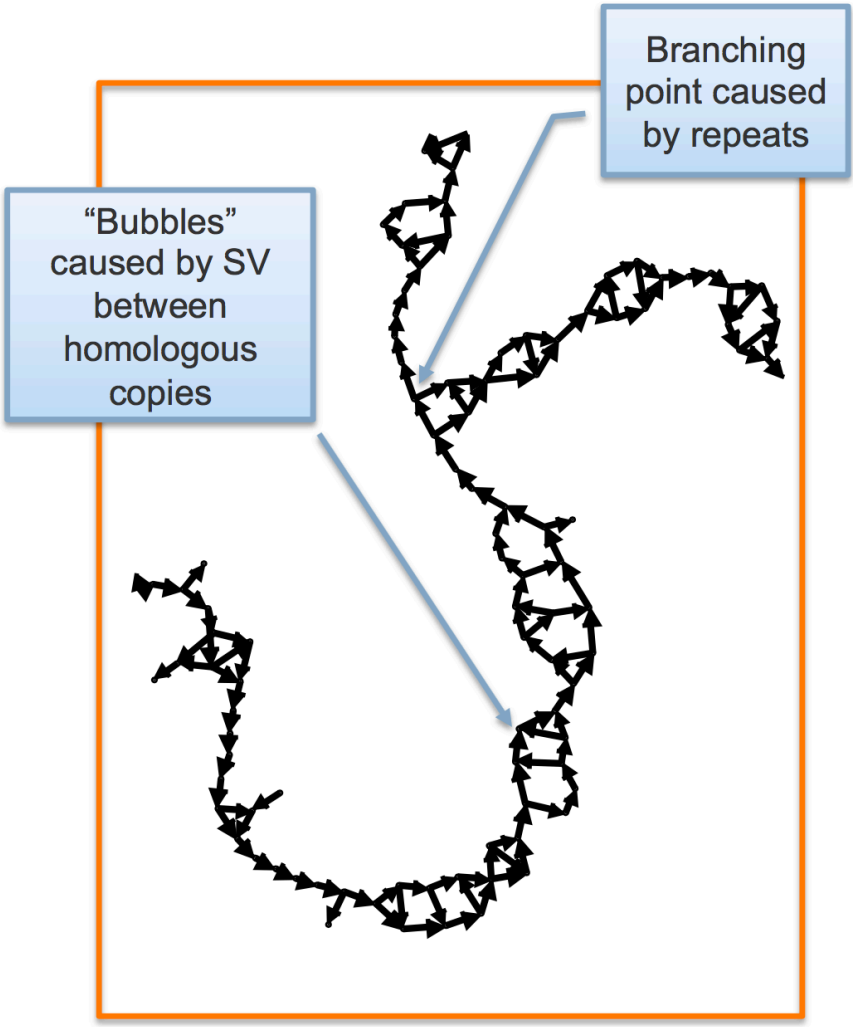
Falcon

SciLifeLab

An unitig graph from Ler-0 + Col-0 data

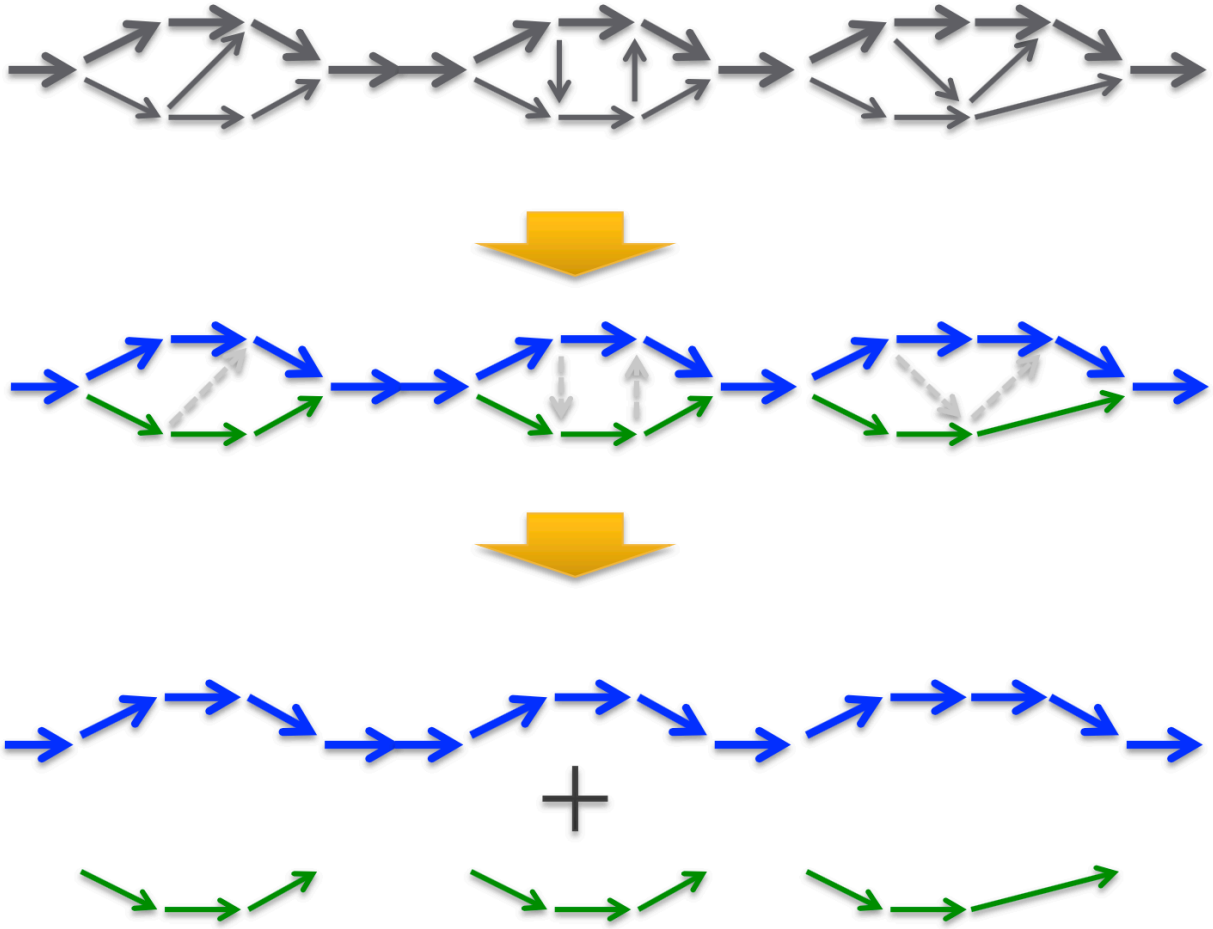


The graph “diameter” ~ 12 M bp
Mean edge size=17.4 k bp



Falcon

SciLifeLab

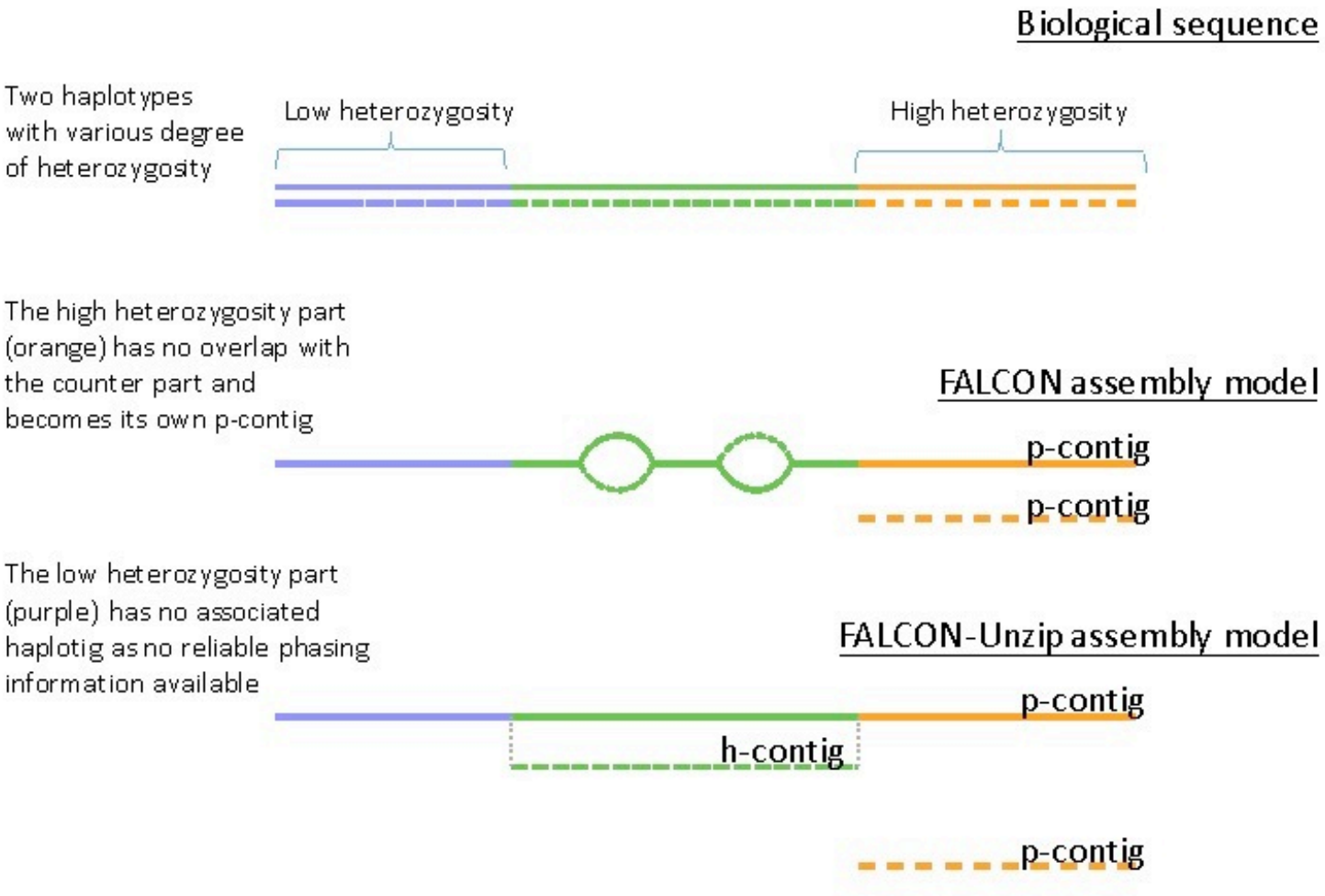


String Bundle

Choose a path
to be the
“primary contig”

Identify
“associated contigs”

Falcon


Biological sequence
Two haplotypes with various degree of heterozygosity
Low heterozygosity
High heterozygosity
The high heterozygosity part (orange) has no overlap with the counter part and becomes its own p-contig
FALCON assembly model
p-contig
p-contig
The low heterozygosity part (purple) has no associated haplotig as no reliable phasing information available
FALCON-Unzip assembly model
p-contig
h-contig
p-contig

Falcon

SciLifeLab

- Running Falcon
  - Make a configuration file
  - Can use SGE queuing manager
  - Run locally on a node
  - Separate filtered subreads into separate fasta's for each movie
    - `zcat *.fastq.gz | seqtk seq -l 5000 -A - | awk 'BEGIN { RS=">"; FS="/" } { print ">"substr($0,1,length($0)-1) > $1".fasta" }'`
  - Make an input fofn
    - `/bin/ls -1 *.fasta > input.fofn`
  - Run Falcon
    - `fc_run.py falcon.cfg`

Falcon

SciLifeLab

- Notes from the author
  - Falcon is limited by file i/o capabilities
    - Lustre file system recommended
    - NFS can handle 3-5 concurrent jobs during pre-assembly
    - Highly repetitive genomes require quadratically more storage space
  - Falcon scales quadratically
    - All-by-all comparison of raw subreads, with matches written to disk

Falcon

SciLifeLab

- The Falcon config file (parameter rich, rest is at end of presentation)

```
[General]
jobtype = local                 # other values sge, slurm
input_fofn = input.fofn
input_type = raw                # uncorrected reads
#input_type = preads            # falcon corrected reads

# The length cutoff used for seed reads used in initial
mapping - these make the corrected reads
length_cutoff = 12000           # use longest 30X coverage

# The length cutoff used for seed reads used for pre-
assembly - the min length of corrected reads
length_cutoff_pr = 12000        # 0-5000 lower than above
```

Miniasm

SciLifeLab

- No error correction step

- Implements Overlap - Layout (but no consensus)

```
# Overlap
minimap/minimap -Sw5 -L100 -m0 -t8 reads.fq reads.fq |
gzip -1 > reads.paf.gz

# Layout
miniasm/miniasm -f reads.fq reads.paf.gz > reads.gfa

# Get fasta
awk '/^S/{print ">"++seq"\n"$3}' reads.gfa > reads.fasta
```

Alternative PacBio Assemblers

SciLifeLab

- ABruijn
  - Uncorrected overlap assembly of long read sequences followed by polishing
  - https://github.com/fenderglass/ABruijn
- Ra
  - Uncorrected overlap assembly of long read sequences
  - https://github.com/mariokostelac/ra-integrate
- ARacon
  - Combination of GraphMap + Miniasm + Racon
  - https://github.com/isovic/aracon
- Hinge
  - Read filtering (but no correction) followed by overlap assembly of long read sequences
  - https://github.com/fxia22/HINGE
- SMARTdenovo
  - Uncorrected overlap assembly of long read sequences
  - https://github.com/ruanjue/smartdenovo

Preliminary Assembly Diagnostics

SciLifeLab

- Assembly Size
  - Assemblathon Script (https://github.com/KorfLab/Assemblathon)
  - Quast

Number of scaffolds	556	
Total size of scaffolds	31318563	
Longest scaffold	447934	
Shortest scaffold	8580	
Number of scaffolds > 1K nt	556	100.0%
Number of scaffolds > 10K nt	555	99.8%
Number of scaffolds > 100K nt	38	6.8%
Number of scaffolds > 1M nt	0	0.0%
Number of scaffolds > 10M nt	0	0.0%
Mean scaffold size	56328	
Median scaffold size	43995	
N50 scaffold length	60037	
L50 scaffold count	152	

Preliminary Assembly Diagnostics

SciLifeLab

- Corrected Read Coverage
  - What happened in the correction process
  - High coverage? Use the ~100X longest subreads

```
--   Found 87386 reads.
--   Found 1654383605 bases (45.95 times coverage).
--
--   Read length histogram (one '*' equals 265.11 reads):
--         0    999      0
--      1000   1999      0
--      2000   2999      0
--      3000   3999      0
--      4000   4999      0
--      5000   5999      0
--      6000   6999      0
--      7000   7999      0
--      8000   8999      0
--      9000   9999      0
--     10000  10999      0
--     11000  11999      0
--     12000  12999      0
--     13000  13999      0
--     14000  14999      0
--     15000  15999  18558 **********************************************************************
--     16000  16999  15099 ********************************************************
--     17000  17999  11974 *********************************************
--     18000  18999   9486 ***********************************
--     19000  19999   7344 ***************************
--     20000  20999   5652 *********************
--     21000  21999   4328 ****************
--     22000  22999   3516 *************
--     23000  23999   2725 **********
--     24000  24999   2057 *******
--     25000  25999   1672 ******
--     26000  26999   1243 ****
--     27000  27999    920 ***
--     28000  28999    735 **
--     29000  29999    541 **
--     30000  30999    414 *
--     31000  31999    324 *
```

Preliminary Assembly Diagnostics

SciLifeLab

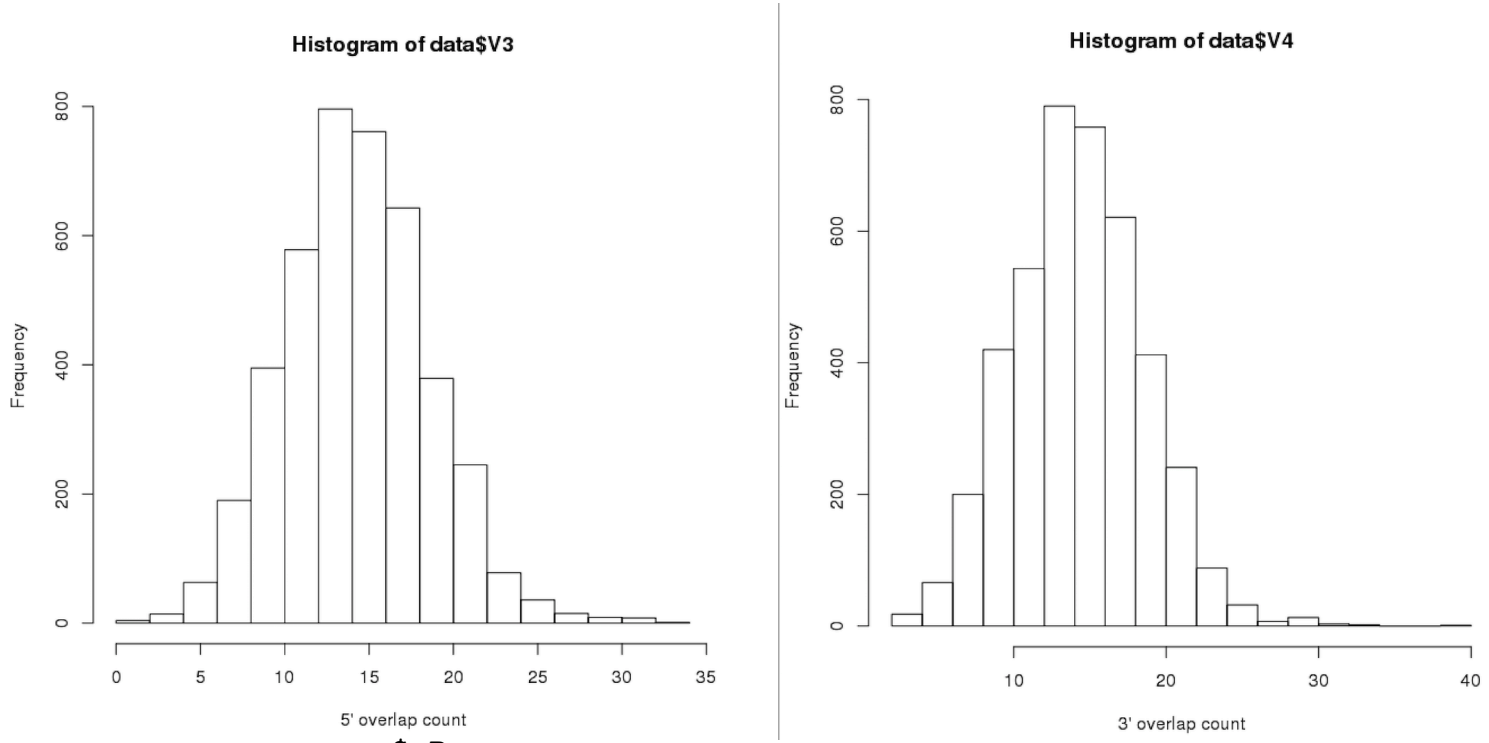
- Falcon: DBstats 1-preads_ovl/preads.db
  - Focus on % Bases column (multiply by read coverage to find cutoff).

```
Statistics for all wells of length 500 bases or more

         12,915 reads        out of          13,124  ( 98.4%)
    116,202,931 base pairs   out of     116,263,784  ( 99.9%)

          8,997 average read length
          6,983 standard deviation

  Base composition: 0.249(A) 0.239(C) 0.258(G) 0.255(T)

  Distribution of Read Lengths (Bin size = 1,000)

        Bin:      Count  % Reads  % Bases     Average
     42,000:          1      0.0      0.0       42279
     41,000:          2      0.0      0.1       41631
...
(more bin values)
...
      3,000:      1,065     75.5     95.0       11317
      2,000:      1,328     85.8     97.9       10259
      1,000:      1,444     97.0     99.7        9251
          0:        387    100.0    100.0        8997
```

Preliminary Assembly Diagnostics

SciLifeLab

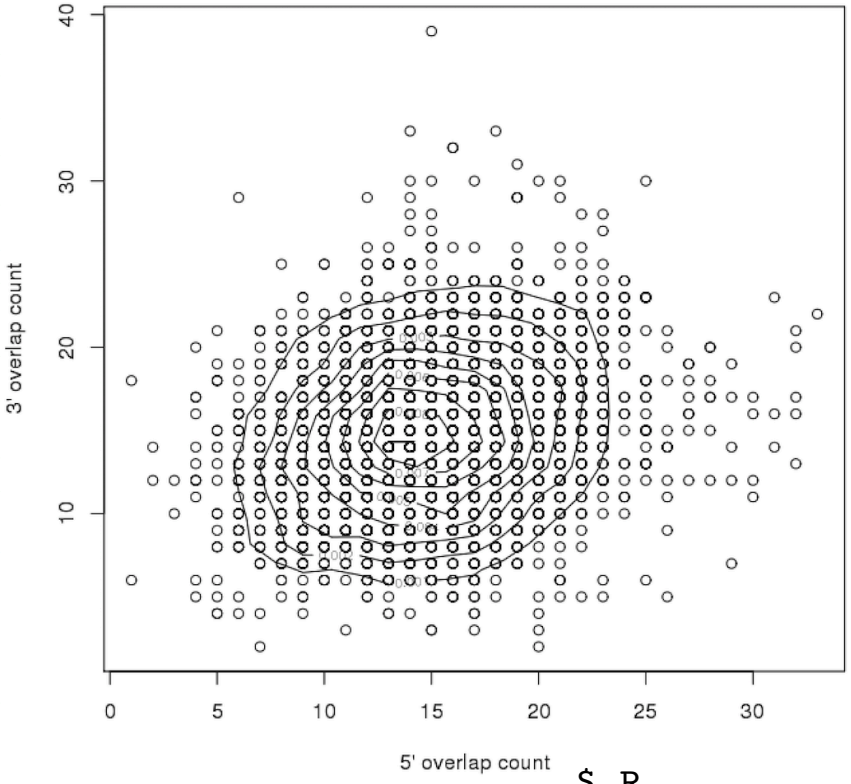
- Falcon: Overlap statistics
  - `cd 1-preads_ovl/ ; fc_ovlp_stats --fofn merge-gather/las.fofn > ovlp_stats.txt`

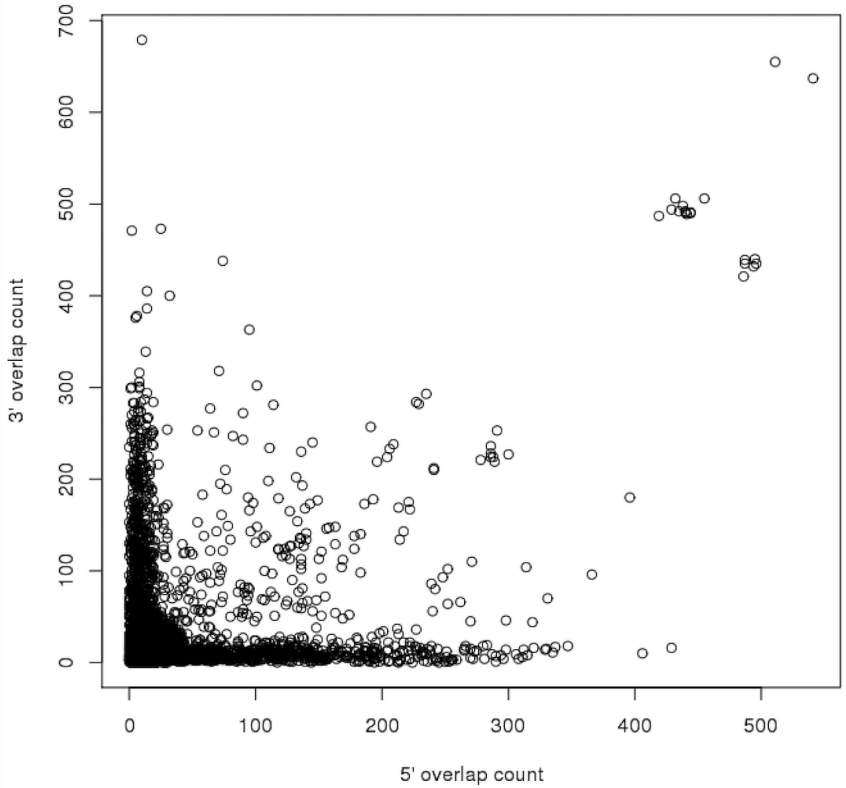



```
$ R
> data <- read.table("ovlp_stats.txt")
> hist(data$V3,xlab="5' overlap")
> hist(data$V4,xlab="3' overlap")
```

Preliminary Assembly Diagnostics

- Falcon: Overlap statistics



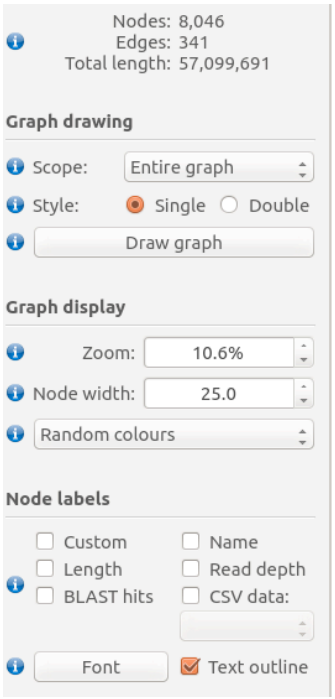


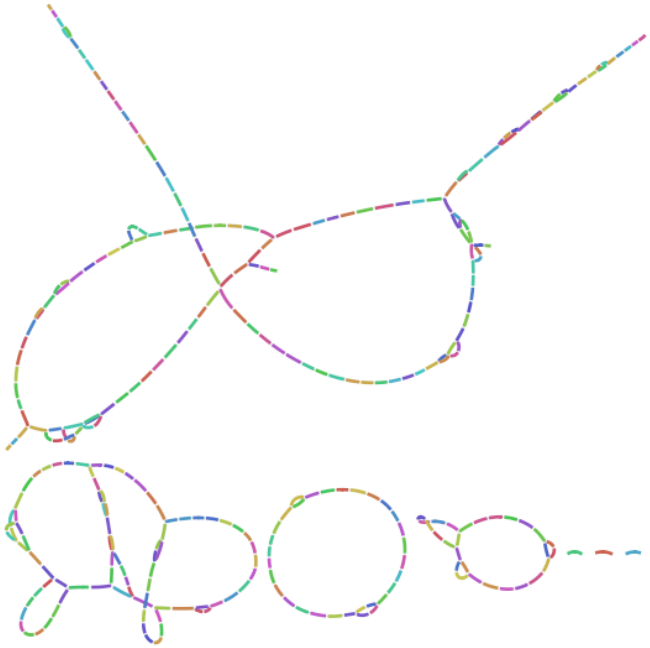

```
$ R
> library(MASS)
> data <- read.table("ovlp_stats.txt")
> plot(data$V3,data$v4)
> z <- kde2d(data$V3,data$v4)
> contour(z,add=TRUE)
```

Preliminary Assembly Diagnostics

SciLifeLab

- Assembly Graph
  - Check connectedness of contigs
    - Is longer range information needed?
      - Higher quality sequence material
      - BioNano
      - Chicago / Dovetail

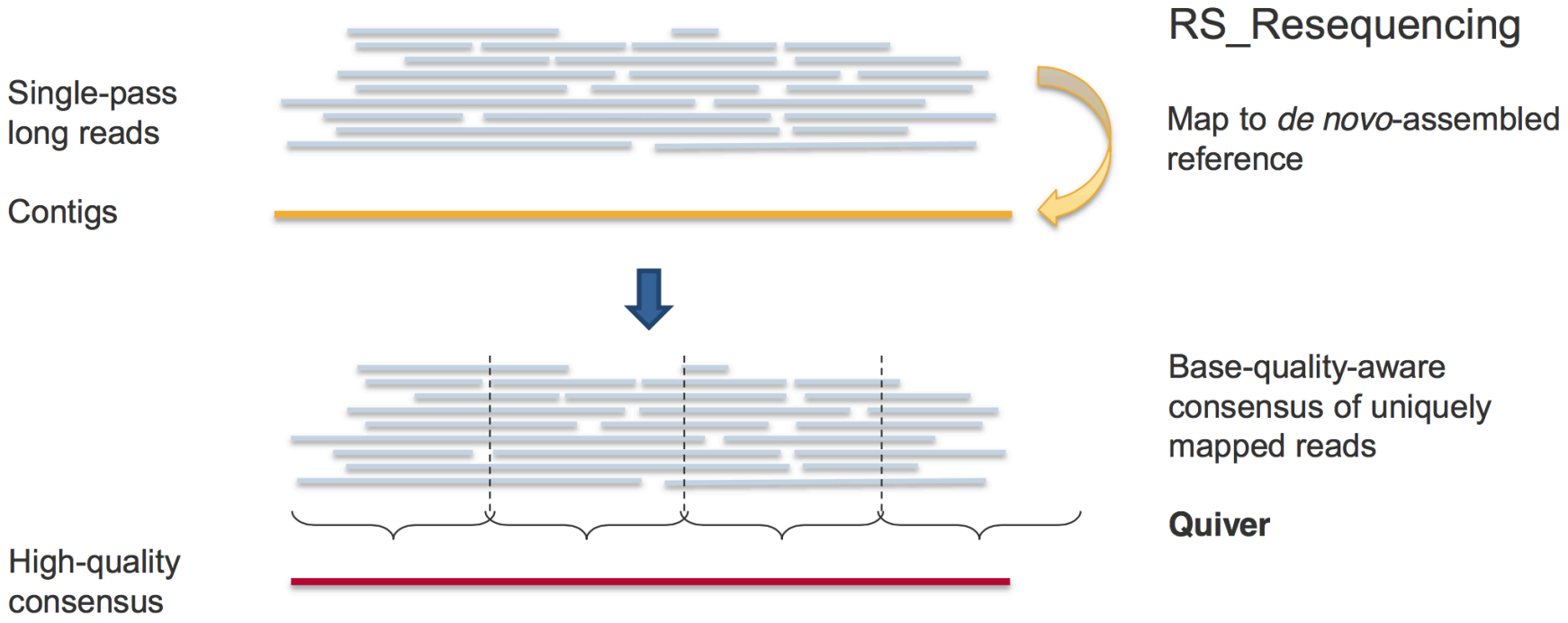




Polishing assemblies

SciLifeLab

- Draft assemblies still contain many InDel and base substitution errors.
  - Correction using Quiver / Arrow and PacBio reads



Polishing assemblies

SciLifeLab

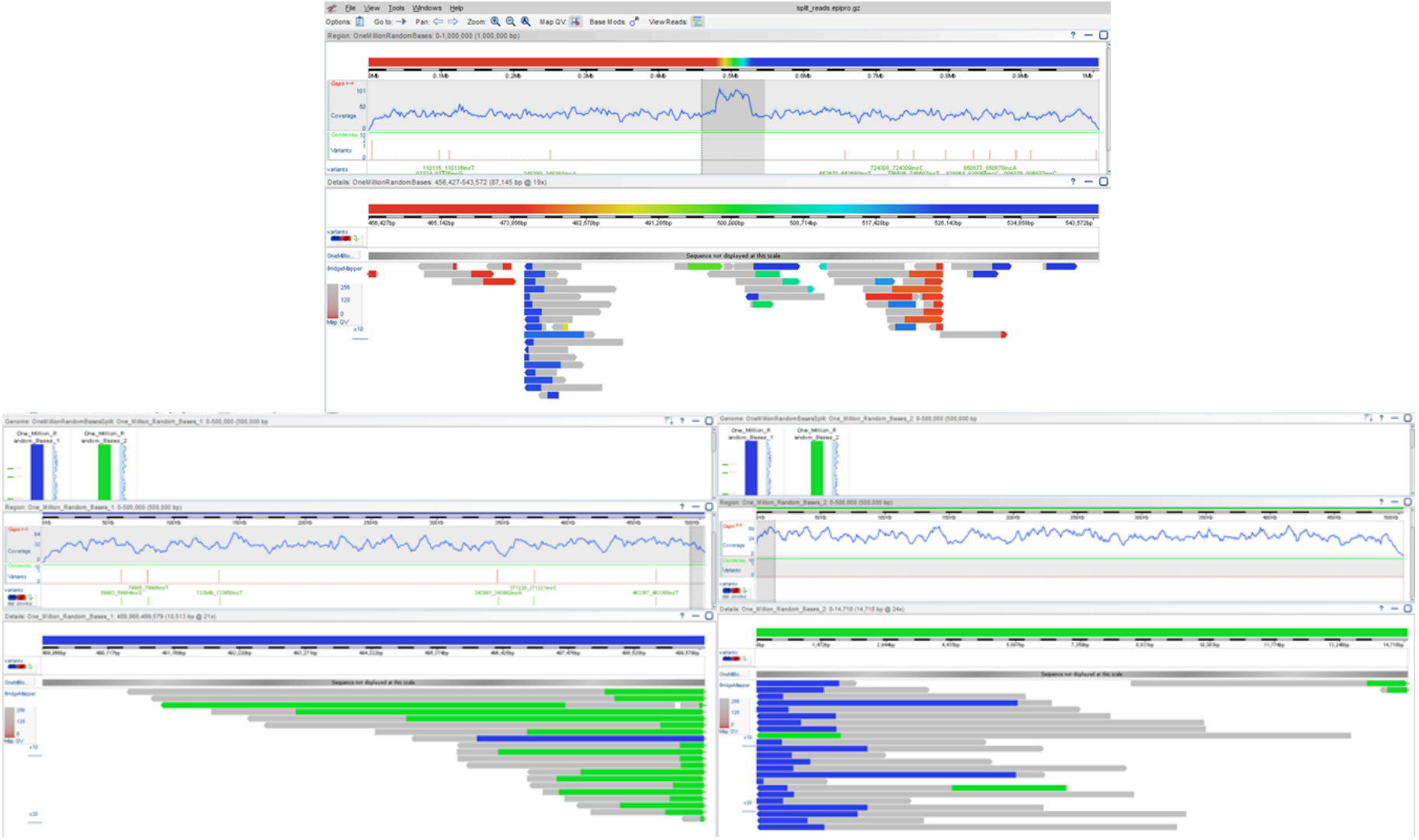
- In the SMRT portal select a protocol that includes the following modules
  - P_Filter
  - P_Mapping
  - P_GenomicConsensus
    - RS_Resequencing
    - RS_BridgeMapper
      - P_BridgeMapper

- To run via SMRT pipe
  - Copy the settings.xml of the dummy job
  - Make the draft assembly a reference using ReferenceUploader
  - Change reference value in settings.xml
  - Run

Polishing assemblies

SciLifeLab

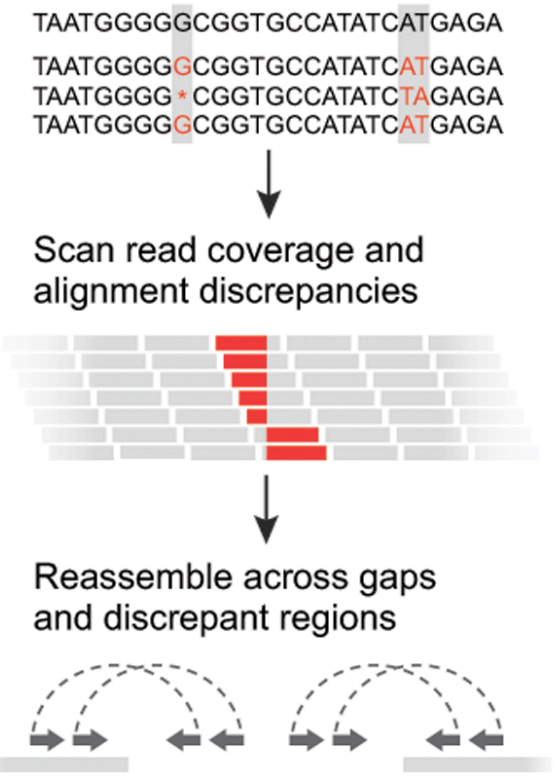
- Bridge Mapper results opened with SMRTview



Polishing assemblies

- Assembly Polishing
  - Can also be performed using Pilon and Illumina reads

PROCESS	RESULT	
Pilon protocol	Assembly improvement (Fasta)	Variation detection (VCF)
Evaluate alignment pileups	Identify and fix base errors	Identify SNPs and small indels
Scan read coverage and alignment discrepancies	Identify potential local misassemblies	Identify larger insertions and deletions
Reassemble across gaps and discrepant regions	Attempt to fill gaps and fix local misassemblies	Attempt to build out the full sequence of larger insertions



Walker et al. 2014. PLOS One

Polishing assemblies

SciLifeLab

- Polishing with Pilon
  - Make an BAM file with your favourite aligner, e.g. BWA.
  - Check ploidy settings

```
java -d64 -Xmx2T -jar pilon-1.16.jar
  --genome unpolished_assembly.fasta
  --frags alignment.bam
  --output polished_assembly
  --vcf --changes --tracks --diploid --threads 48
```

Summary

SciLifeLab

- PacBio sequencing is very dependent on sample DNA quality
- The longest reads are targeted for correction
- Correction with Illumina only does part of the read correction job
- Check basic stats
- Select your best assemblies
- Polish
- Assess correctness.

Falcon

SciLifeLab

- The Falcon config file

```
[General]
jobtype = local                 # other values sge, slurm
input_fofn = input.fofn
input_type = raw                # uncorrected reads
#input_type = preads            # falcon corrected reads

# The length cutoff used for seed reads used in initial
mapping - these make the corrected reads
length_cutoff = 12000           # use longest 30X coverage

# The length cutoff used for seed reads used for pre-
assembly - the min length of corrected reads
length_cutoff_pr = 12000        # 0-5000 lower than above
```

Falcon

SciLifeLab

- The Falcon config file cont'd.

```
# concurrency settings
pa_concurrent_jobs = 32       # pre-assembly
ovlp_concurrent_jobs = 32     # overlap
cns_concurrent_jobs = 32      # consensus

# overlapping options for Daligner
pa_HPCdaligner_option = -dal4 -t16 -e.70 -l1000 -s1000
ovlp_HPCdaligner_option = -dal4 -t32 -h60 -e.96 -l500 -
s1000

# -B <int>, -dal <int>
# blocks to compare => higher = less but longer jobs

# -e <int>   # average correlation rate (def 70%)
```

Falcon

SciLifeLab

- The Falcon config file cont'd.

```
# -v                                    # turns on verbose

# -l <int>
# the length in base pairs of the minimum local
alignment (def. 1000)

# -s <int>
# how frequently trace alignments measured in bases are
recorded (def. 100)

# -b
# daligner assumes the data has a strong compositional
bias (e.g. >65% AT rich).
```

Falcon

SciLifeLab

- The falcon config file cont'd.

```
# -t <int>,-M <int>          # Limits the effects of repeats

# Invariably, some k-mers are significantly over-
represented (e.g. homopolymer runs). These k-mers create an
excessive number of matching k-mer pairs and left
unaddressed would cause daligner to overflow the available
physical memory.  One way to deal with this is to
explicitly set the -t parameter which suppresses the use of
any k-mer that occurs more than t times in either the
subject or target block.  However, a better way to handle
the situation is to let the program automatically select a
value of t that meets a given memory usage limit specified
(in Gb) by the -M parameter.  By default daligner will use
the amount of physical memory as the choice for -M.  If you
want to use less, say only 8Gb on a 24Gb HPC cluster node
because you want to run 3 daligner jobs on the node, then
specify -M8.  Specifying -M0 basically indicates that you
do not want daligner to self adjust k-mer suppression to
fit within a given amount of memory.
```

Falcon

SciLifeLab

- The falcon config file cont'd.

```
# -H <int>
# By default daligner compares all overlaps between
reads in the database that are greater than the minimum
cutoff set when the DB or DBs were split, typically 1 or
2 Kbp.  However, the HGAP assembly pipeline only wants
to correct large reads, say 8Kbp or over, and so needs
only the overlaps where the a-read is one of the large
reads.  By setting the -H parameter to say N, one alters
daligner so that it only reports overlaps where the a-
read is over N base-pairs long.

# Essentially limits making alignments of reads of any
size only to reads longer than <int>
```

Falcon

SciLifeLab

- The Falcon config file cont'd.

```
# -k <int>, -h <int>, -w <int>
# The options -k, -h, and -w control the initial
filtration search for possible matches between reads.
Specifically, the daligner search code looks for a pair
of diagonal bands of width 2^w (default 2^6 = 64) that
contain a collection of exact matching k-mers (default
14) between the two reads, such that the total number of
bases covered by the k-mer hits is h (default 35). k
cannot be larger than 32 in the current implementation.
```

Falcon

SciLifeLab

- The Falcon config file cont'd.

```
# How the database is split up for making comparison
blocks
pa_DBspliAt_option = -x1000 -s50 -a
ovlp_DBsplit_option = -x1000 -s50 -a


# -x <int>
# Ignore reads lower than length

# -s <int>
# specifies number of mb in each DB chunk - larger
numbers makes smaller numbers of longer jobs (should be
400 mb or so for large genomes)

# -a        # ignore secondary reads from the same well
```

Falcon

SciLifeLab

- The Falcon config file cont'd.

```
# error correction consensus option
falcon_sense_option = --output_multi --min_idt 0.70 --
min_cov 4 --local_match_count_threshold 2 --max_n_read
200 --n_core 6


# --min_cov <int>
# break/trim seed read lower than <int>

# --max_n_read <int>
# max reads used for error correction - reduce value for
highly repetitive genomes
```

Falcon

SciLifeLab

- The Falcon config file cont'd.

```
# overlap filtering options
overlap_filtering_setting = --max_diff 100 --max_cov 100
--min_cov 20 --bestn 10

# --bestn <int>
# Use the <int> best overlaps to simplify transitive
edges in the graph

# --max_cov <int>, --min_cov <int>
# filter overlaps that are too high or too low (e.g.
reads ending in repeats, or many sequencing errors)

# --max_diff <int>
# Max difference of coverage between 5' and 3' ends
```