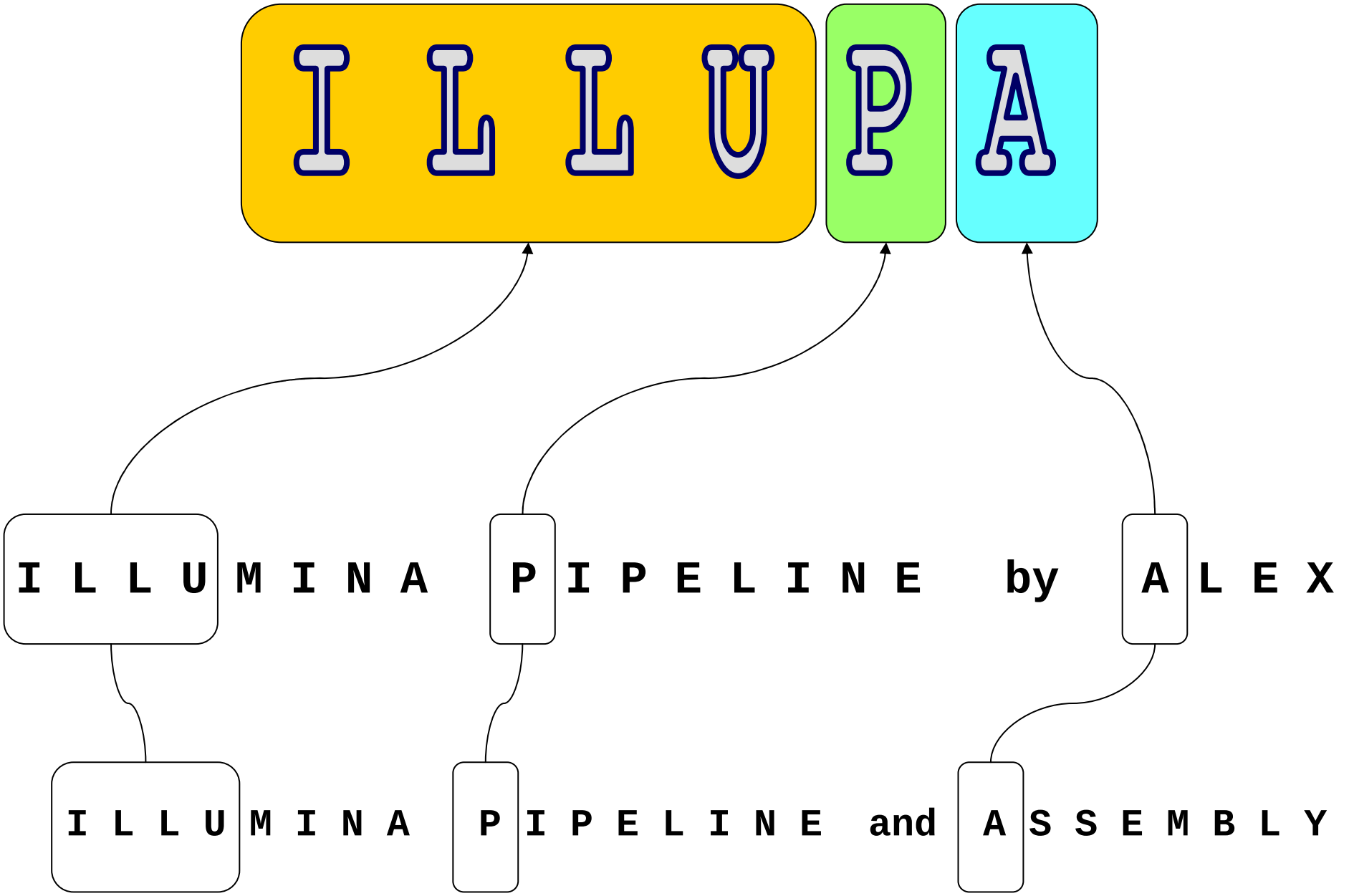


January 2010

OUTLINE:

project overview and dataflow

IGA output files

IGA quality scores

filtering for high quality reads

assembly parameters

visualization with Python Contig Viewer

Illumina GA2 high-throughput sequencing

Current performance:

10 to 20 million reads per lane

8 lanes per chip (flow cell)

single read: 85 to 125 nt

paired reads: 125 <-> 85+

80 to 160 million reads per chip

Conservative calculations: 100 M reads X 100 nt length = 10 Gb per run

Computational Facilities at the UCD Genome Center

- **Storage arrays - 48 Tb HD space**
- **Computational servers with up to 128 and 512 Gb of RAM**
- **Computational clusters with 100+ nodes**

Illumina sequencing dataflow

Illumina raw reads on sequencing machine

rsync/sftp file transfer

processed reads on file server (storage array)

QC and filtering for high quality reads

input files for downstream assembly and analysis

assembly using CLC and/or Velvet

de novo assembled contigs

analysis and annotation of assemblies

data distribution via sftp or web servers

Project Goals

lettuce transcriptome sequencing

lettuce gene space sequencing

...

lettuce whole genome sequencing (~3 Gb)

Current Sequencing Scale Estimate

Transcriptome - 4 to 6 lanes sufficient (~80 million reads)

Gene space - 2 to 4 chips required (at leads 200 million reads)

project overview and dataflow

IGA output files

IGA quality scores

filtering for high quality reads

assembly parameters

visualization with Python Contig Viewer

Illumina Pipeline 1.5 - real time analysis

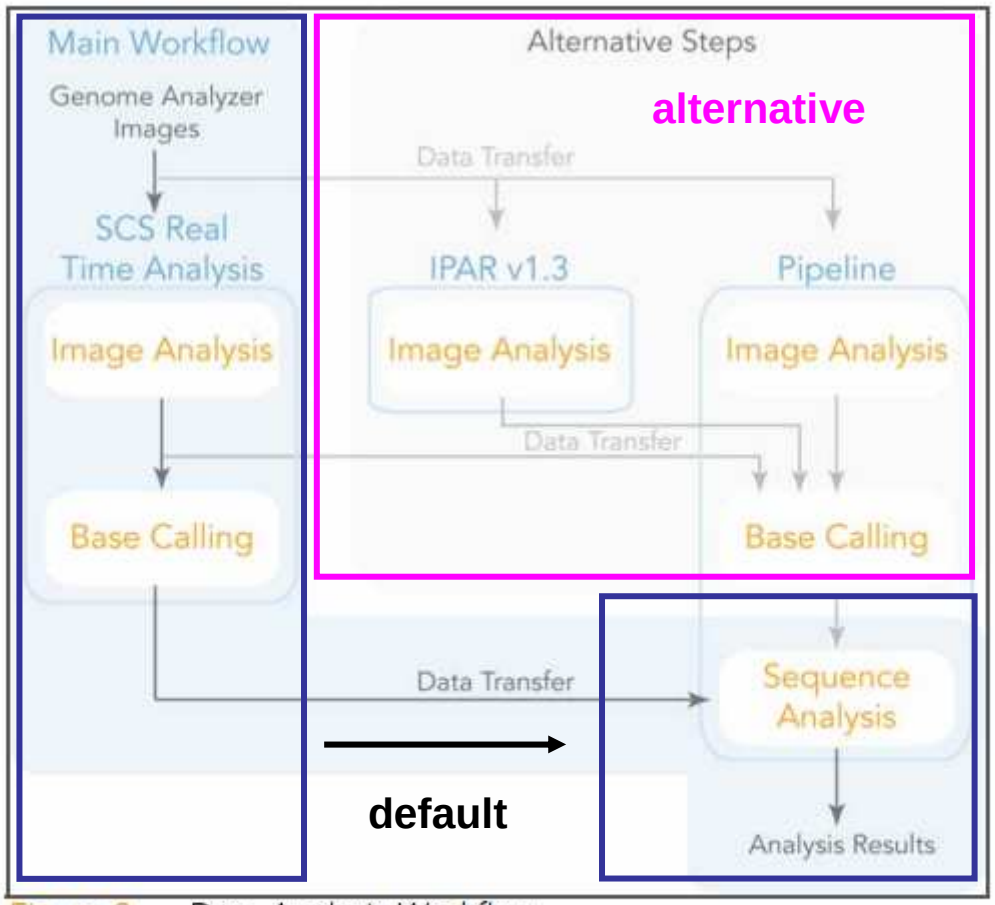


Figure 2 *Data Analysis Workflow*

The standard workflow is to perform image analysis and base calling using SCS real time analysis, after which Pipeline performs alignment using the base calling results (Figure 2).

Figure 6 illustrates a typical Run Folder after IPAR image analysis and Pipeline base calling and alignment, or one containing only Pipeline analysis data.

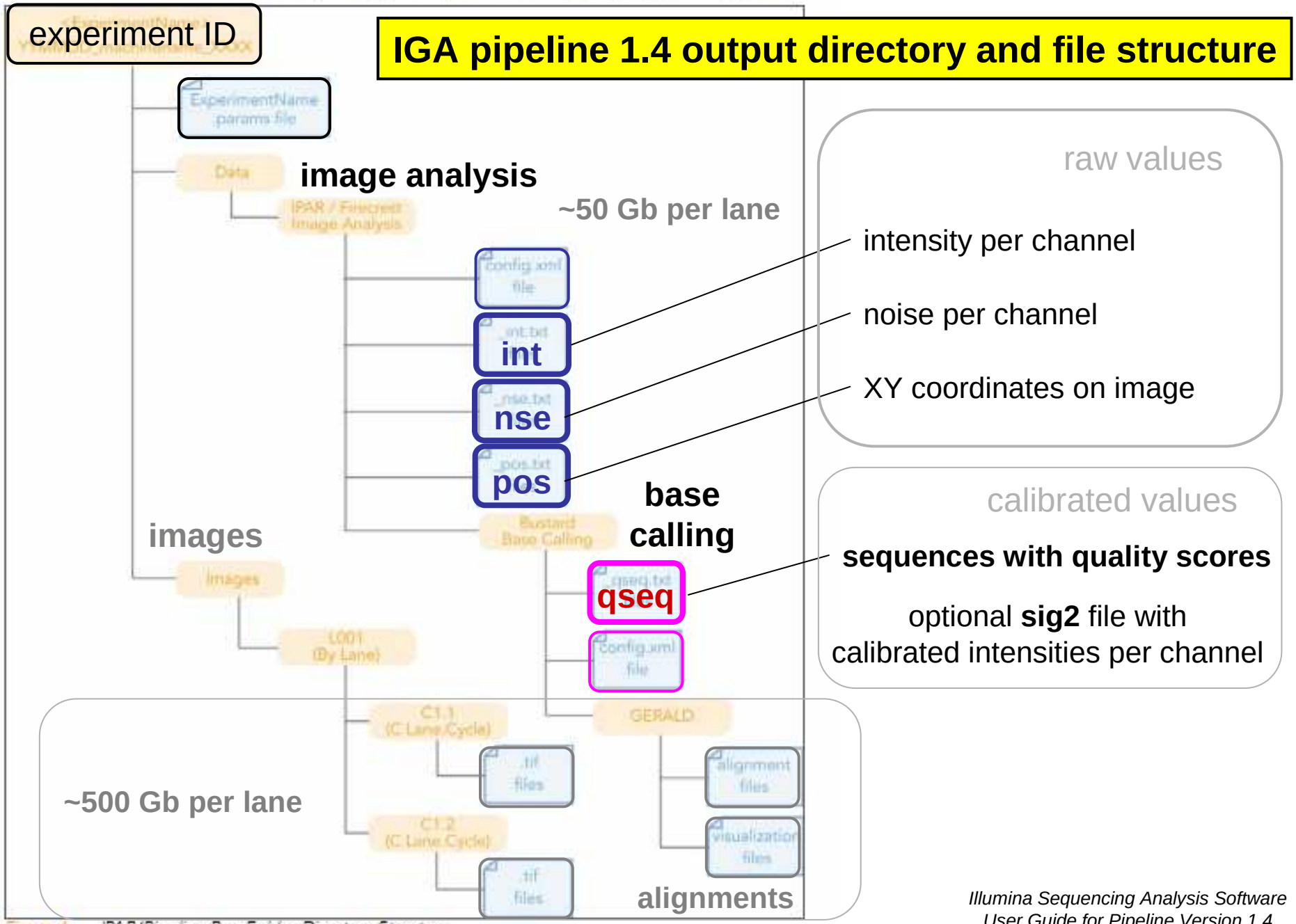


Figure 6 IPAR/Pipeline Run Folder Directory Structure

Pipeline 1.5 output directory structure

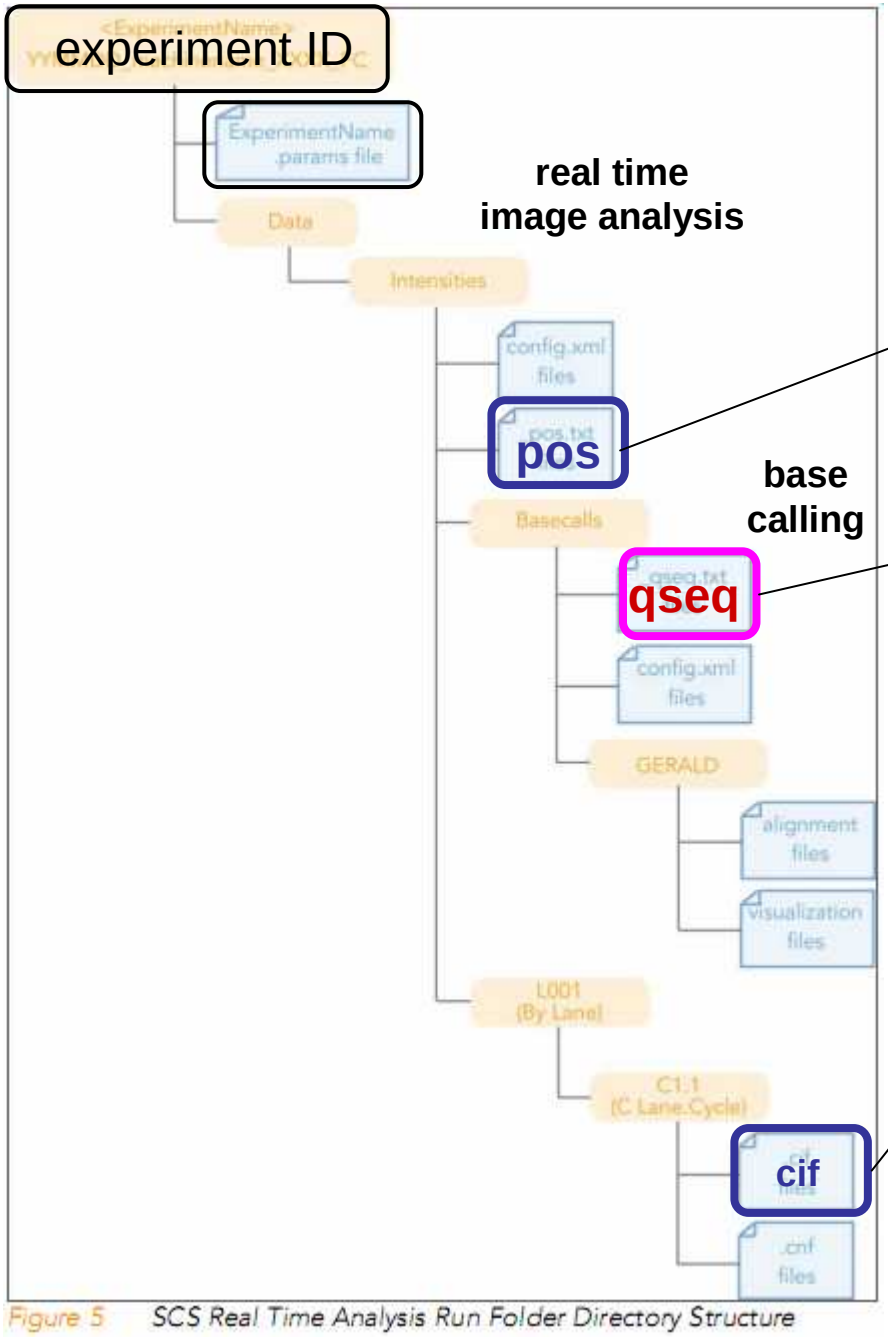


Figure 5 SCS Real Time Analysis Run Folder Directory Structure

XY coordinates on image

calibrated values

sequences with quality scores

The Pipeline operates in a specific directory called the Run Folder where the images and analysis output files are saved by default in a consistent hierarchical structure. A Run Folder containing SCS real time analysis data is very similar to a Run Folder containing IPAR analysis data, or a Run Folder containing only Pipeline analysis data

intensity per channel in binary format

these are not available anymore:

- int — intensity per channel
- nse — noise per channel

ILLUMINA Filtering Options for HQ reads

To remove the least reliable data from the analysis, the raw data can be filtered to remove any clusters that have “too much” intensity corresponding to bases other than the called base. By default, the purity of the signal from each cluster is examined over the first 25 cycles and chastity is calculated for each cycle:

chastity = Highest_Intensity / (Highest_Intensity + Next_Highest_Intensity)

Chastity:

the ratio of the brightest intensity over the sum of the brightest and second brightest intensities

Filtering Options:

failed-chastity: the number of bases where the chastity is lower than the given threshold (1)

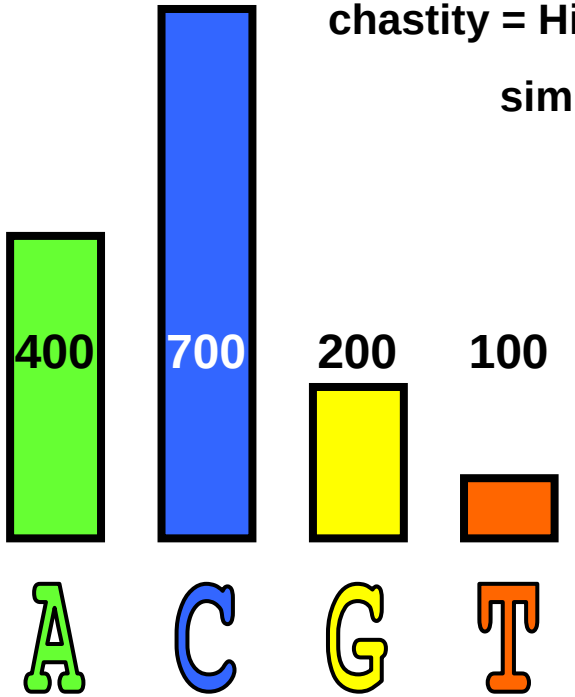
chastity: the minimum chastity (0.6)

purity: the minimum purity (25)

By default, the pipeline filters the clusters according to the relation "failed-chastity<=1", using a chastity threshold of 0.6, on the first 25 cycles. This removes all clusters with a chastity less than 0.6 on two or more bases among the first 25 bases.

```
--smt-filter failed-chastity --smt-relation le (less or equal)
--smt-threshold 1 --chastity-threshold 0.6
--pure-bases 25
```

[The new default filtering implemented in Bustard is that at most one cycle is less than the chastity threshold]



chastity = Highest_Intensity / (Highest_Intensity + Next_Highest_Intensity)

simple_ratio = Next_Highest_Intensity / Highest_Intensity

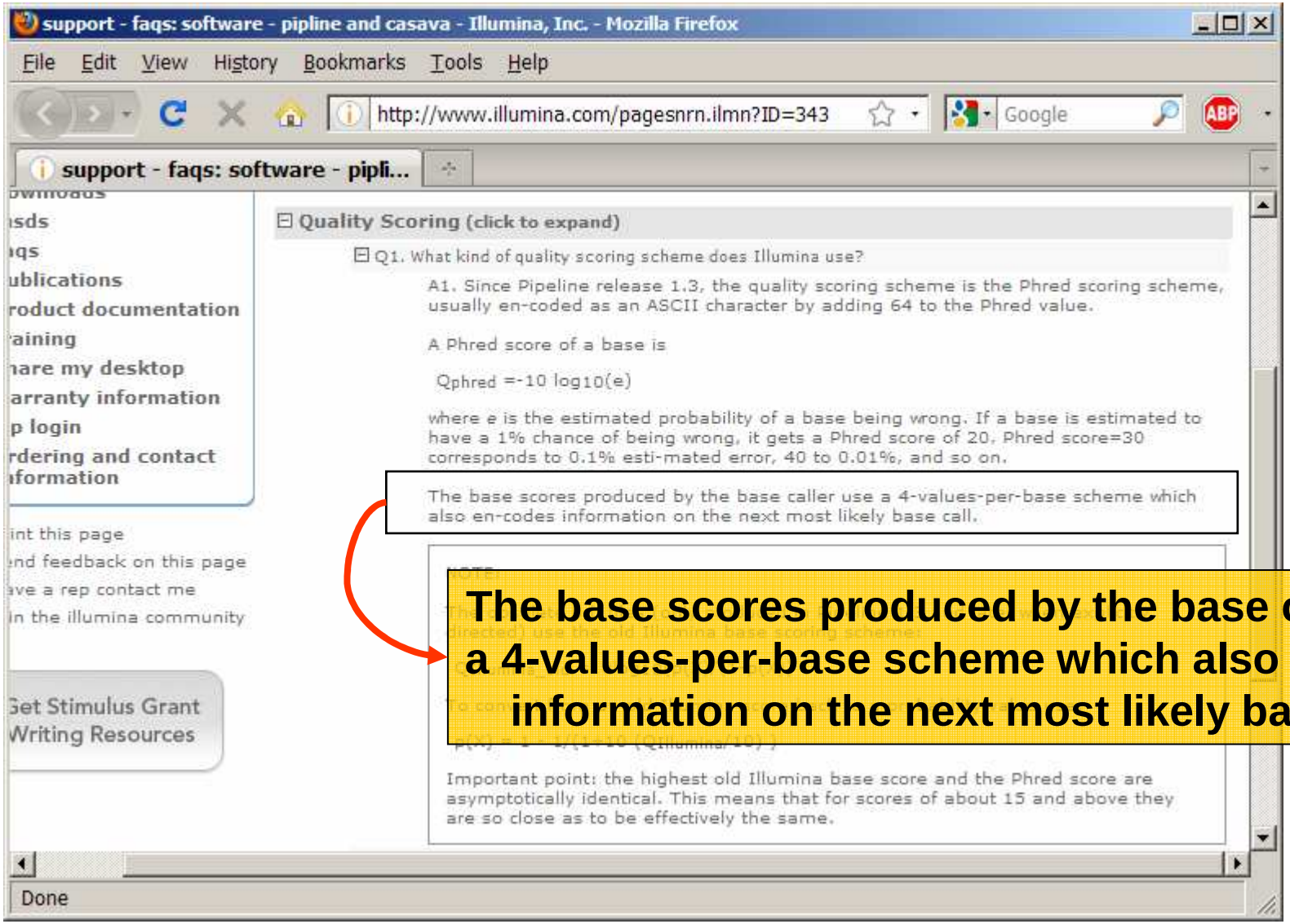
quality categories for simple ratio:

```
A  -  [  0.0  < Q <=  0.2  ]
B  -  [  0.2  < Q <=  0.4  ]
C  -  [  0.4  < Q <=  0.6  ]
D  -  [  0.6  < Q <=  0.8  ]
F  -  [  0.8  < Q <=  1.0  ]
```

Intensity 1-st	1000	1000	1000	1000	1000	1000	1000
Intensity 2-nd	100	200	300	400	600	800	900
IGA chastity	**0.91**	**0.83**	**0.77**	**0.71**	**0.63**	**0.56**	**0.53**
(1.0 - ratio)	0.90	0.80	0.70	0.60	0.40	0.20	0.10
ratio	**0.1**	**0.2**	**0.3**	**0.4**	**0.6**	**0.8**	**0.9**
quality category	**A** (I)	**A** (I)	**B** (9)	**B** (9)	**C** (5)	**D** (0)	**F** (#)

ILLUMINA Quality scores

http://www.illumina.com/pagesnrn.ilmn?ID=343



The base scores produced by the base caller use a 4-values-per-base scheme which also en-codes information on the next most likely base call.

A quality value Q is an integer mapping of p, the probability of the corresponding sequence letter being incorrect.
It is calculated as follows:

http://en.wikipedia.org/wiki/Phred_quality_score

$$Q_{\text{phred}} = -10 \log_{10} p$$

Binary	Oct	Dec	Hex	Glyph	
100 0010	102	66	42	B	2
100 0011	103	67	43	C	3
100 0100	104	68	44	D	
100 0101	105	69	45	E	
100 0110	106	70	46	F	
100 0111	107	71	47	G	
100 1000	110	72	48	H	
100 1001	111	73	49	I	
100 1010	112	74	4A	J	10
100 1011	113	75	4B	K	
100 1100	114	76	4C	L	
100 1101	115	77	4D	M	
100 1110	116	78	4E	N	
100 1111	117	79	4F	O	
101 0000	120	80	50	P	
101 0001	121	81	51	Q	
101 0010	122	82	52	R	
101 0011	123	83	53	S	
101 0100	124	84	54	T	20
101 0101	125	85	55	U	
101 0110	126	86	56	V	
101 0111	127	87	57	W	
101 1000	130	88	58	X	
101 1001	131	89	59	Y	
101 1010	132	90	5A	Z	
101 1011	133	91	5B	[	
101 1100	134	92	5C	\	
101 1101	135	93	5D	]	
101 1110	136	94	5E	^	30
101 1111	137	95	5F	_	
110 0000	140	96	60	`	
110 0001	141	97	61	a	
110 0010	142	98	62	b	

http://en.wikipedia.org/wiki/FASTQ_format

A quality value Q is an integer mapping of p, the probability of the corresponding sequence letter being incorrect.
It is calculated as follows:

$$Q_{\text{phred}} = -10 \log_{10} p$$

low quality

$-10*\log_{10}(0.5) = -10 * (-0.3) = 3$

$-10*\log_{10}(0.1) = -10 * (-1) = 10$

medium quality

$-10*\log_{10}(0.01) = -10 * (-2) = 20$

http://en.wikipedia.org/wiki/ASCII

Quality String - in symbolic ASCII format (ASCII chracter code = quality value + 64)

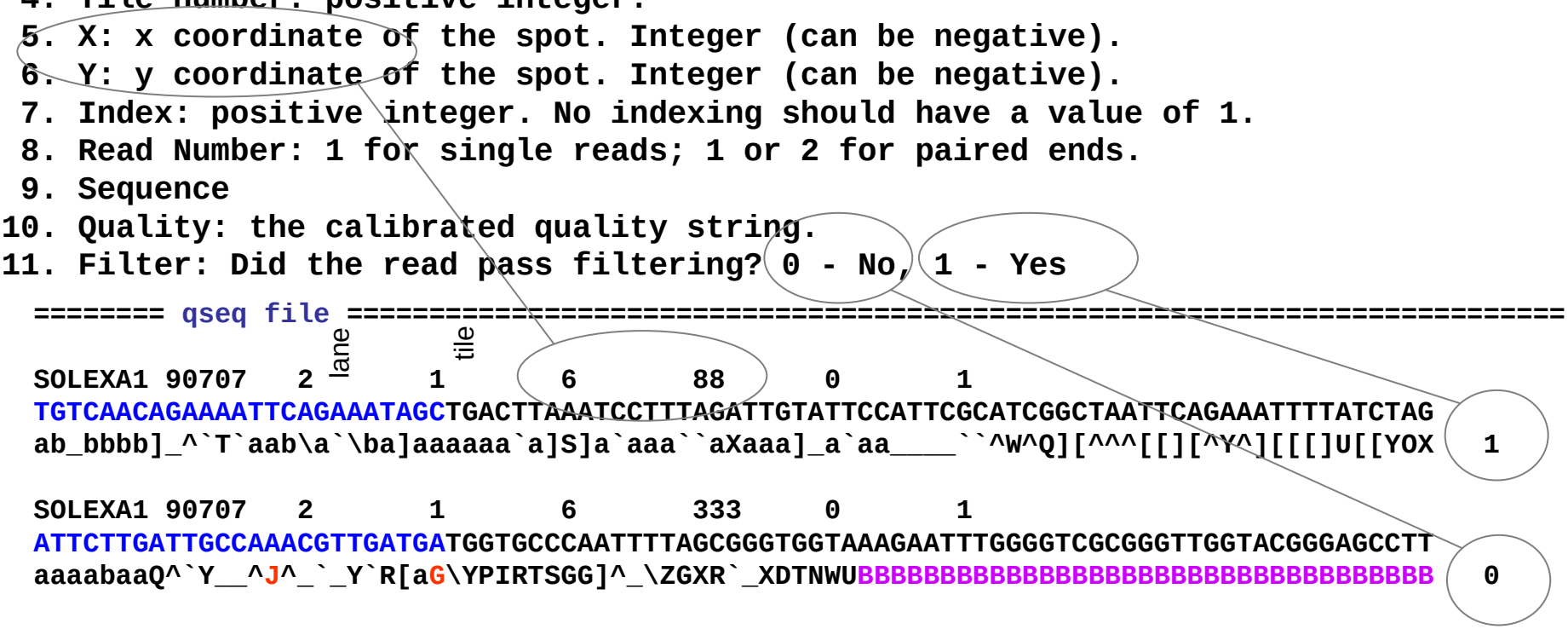
high quality

$-10*\log_{10}(0.001) = -10 * (-3) = 30$

Illumina 1.3+ format can encode a Phred quality score from 0 to 62 using ASCII 64 to 126 (although in raw read data Phred scores from 0 to 40 only are expected).

The main output files in Bustard are the *_qseq files. They have the following format:

1. Machine name: (hopefully) unique identifier of the sequencer.
2. Run number: (hopefully) unique number to identify the run on the sequencer.
3. Lane number: positive integer (currently 1-8).
4. Tile number: positive integer.
5. X: x coordinate of the spot. Integer (can be negative).
6. Y: y coordinate of the spot. Integer (can be negative).
7. Index: positive integer. No indexing should have a value of 1.
8. Read Number: 1 for single reads; 1 or 2 for paired ends.
9. Sequence
10. Quality: the calibrated quality string.
11. Filter: Did the read pass filtering? 0 - No, 1 - Yes




```
======== qseq file ============================================================

SOLEXA1 90707   2       1       6       88      0       1
TGTCAACAGAAAATTCAGAAATAGCTGACTTAAATCCTTTAGATTGTATTCCATTCGCATCGGCTAATTCAGAAATTTTATCTAG
ab_bbbb]_^`T`aab\a`\ba]aaaaaa`a]S]a`aaa``aXaaa]_a`aa____``^W^Q][^^^[][^Y^][[]U[[YOX   1

SOLEXA1 90707   2       1       6       333     0       1
ATTCTTGATTGCCAAACGTTGATGATGGTGCCCAATTTTAGCGGGTGGTAAAGAATTTGGGGTCGCGGGTTGGTACGGGAGCCTT
aaaabaaQ^`Y__^J^_`_Y`R[aG\YPIRTSGG]^_\ZGXR`_XDTNWUBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB   0

================================================================================

FASTA file:
>90707:SOLEXA1:2:1:6:88#0/1   891  LEN: 85
TGTCAACAGAAAATTCAGAAATAGCTGACTTAAATCCTTTAGATTGTATTCCATTCGCATCGGCTAATTCAGAAATTTTATCTAG

FASTQ file:
@90707:SOLEXA1:2:1:6:88#0/1
TGTCAACAGAAAATTCAGAAATAGCTGACTTAAATCCTTTAGATTGTATTCCATTCGCATCGGCTAATTCAGAAATTTTATCTAG
+90707:SOLEXA1:2:1:6:88#0/1
ab_bbbb]_^`T`aab\a`\ba]aaaaaa`a]S]a`aaa``aXaaa]_a`aa____``^W^Q][^^^[][^Y^][[]U[[YOX
```

project overview and dataflow

IGA output files

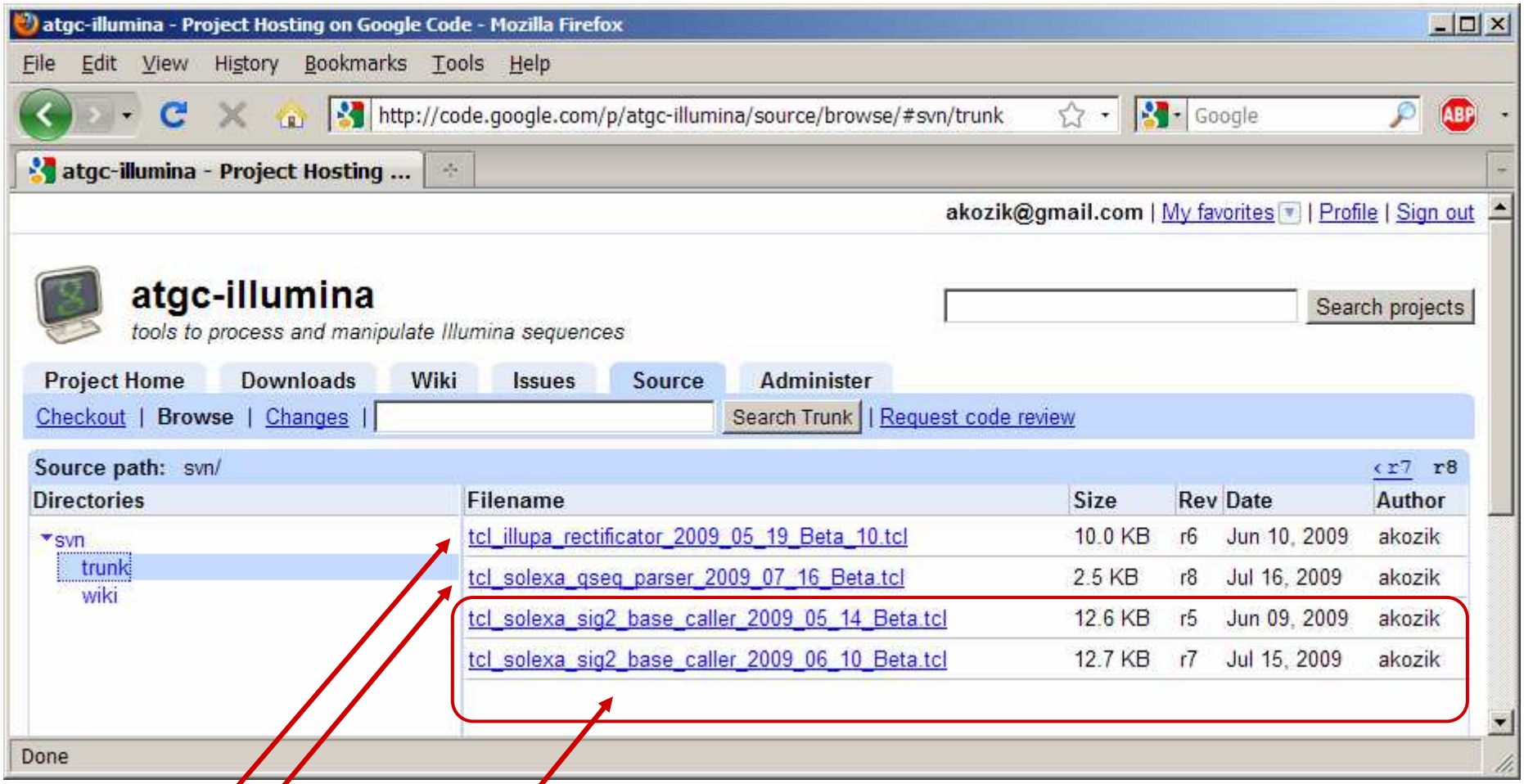
IGA quality scores

filtering for high quality reads

assembly parameters

visualization with Python Contig Viewer

http://code.google.com/p/atgc-illumina/



sig2 basecalling / filtering (obsolete ?)

qseq parser - it takes into account IGA quality scores

illupa rectificator - to transform raw intensities in vertical format into linear sig2 like order

qseq filtering approach summary:

qseq parser analyzes data from qseq files

filtering criteria ‘1’

```
SOLEXA1 90707   8       120     6       1512    0       1
AAAAACAGATGCTACTTACTTTACTACCATATTTCCCAAAACAAAGAGCTAACAGTAGAACATGACGC.C.TAACATTCTCATGA
a`aab_Z_V__aXaa\[aa``^a`[^]ab`aa^aa_a]SXZZ]`UXXS_Y`__^^S^^Y^S^GU]^]TDIDL]SSZYPWU[[[V[   1
```

sequence after first failed call is removed

```
>90707:SOLEXA1:8:120:6:1512#0/1   691  LEN: 68
AAAAACAGATGCTACTTACTTTACTACCATATTTCCCAAAACAAAGAGCTAACAGTAGAACATGACGC
```

future plans - to analyze string of quality scores regardless of IGA default filtering criteria

qseq parser output example - *.trim.fa file

```
#!/usr/bin/tcl

proc Process_Qseq {argv} {

    # set length_cut 60
    length_cut 40


    set f_in1  [open [lindex $argv 0] "r"]
    set f_out  [open [lindex $argv 1] "w"]
    ....
    set tag_l  [lindex $argv 2]
    set r_len  [lindex $argv 3]
```

```
bash-2.03$ tclsh8.3 tcl_solexa_qseq_parser_2009_07_16_Beta.tcl
Program usage:
qseq_file_to_process, output_file, tag_length, read_length
bash-2.03$
```

Without Tag:

```
>90707:SOLEXA1:8:120:6:1333#0/1   692  LEN: 68
GAATTCCTTCGCACATCATCATAAACCTCGGTTCCTCCATTTGTCAAAAAGCAAACGGGGAATTTTTG

>90707:SOLEXA1:8:120:6:1245#0/1   693  LEN: 68
GGGTGAATGTAAAGGAATATGAGGCTCTTCGACCTGACAAGCTTGCTGTGGATGGAGGCGAACAGAAT

>90707:SOLEXA1:8:120:6:236#0/1   694  LEN: 85
CGGCGATCTGATTCCGATGAGCAGTCAACCGCAAGCTCCACAGAGTAGGAGCCTGGTTATTACGGCTGGCGCAGGCGGAACCAAA

>90707:SOLEXA1:8:120:6:98#0/1   696  LEN: 85
ATCAGCTTTCCATGTAGCATAATAGTAACGCTGCCTAAGAAGTTGCAAATTCTCCCTGTCTAGAGAAACTGGCTGAGAGCCTGGA
```

With Tag (bar code):

```
>PEFC42D7CAAXX:SOLEXA1:2:1:6:980#0/1  [ADPT: _GCTACAT_]   404  LEN: 54
ATAATAAAATACTTCATATCTATGTGTAAGATTAAAGGGACCATGCACTCACCT

>PEFC42D7CAAXX:SOLEXA1:2:1:6:509#0/1  [ADPT: _GCATAGT_]   405  LEN: 54
CATCATCTTGATCTTTTGCTCCACTTGGCCCTGCTCCATCATCTTGTGGATGCC

>PEFC42D7CAAXX:SOLEXA1:2:1:6:1127#0/1  [ADPT: _ACTAGCT_]   408  LEN: 54
TCCCTCCTTTAAAACACCCCAAATTGCAATTTAACAATACTTGAGACTTTGGGC
```

sig2 file example

lane	tile	X coordinate	Y coordinate	calibrated intensity scores per channel											
				1-st cycle				2-nd cycle				3-d cycle			
				A	C	G	T	A	C	G	T	A	C	G	T
8	120	5	1105	-249	44	523	315	-248	79	25	443	-146	-138	146	535
8	120	5	942	-103	164	486	26	-25	598	-59	-34	-77	564	18	43
8	120	5	1392	-67	97	501	1	-101	234	-40	154	-128	-101	306	238
8	120	5	1114	658	62	61	52	711	165	43	-46	646	13	-1	101
8	120	5	1409	-124	424	227	-82	548	-92	-19	-99	210	25	98	26
8	120	5	327	-18	-69	485	-24	-39	355	-31	-72	-173	450	-14	-3
8	120	5	1017	-149	587	144	15	-259	14	-12	697	-67	472	1	161
8	120	5	413	476	-23	-44	40	-97	88	17	429	-95	56	-53	493
8	120	5	453	-99	661	-82	-37	621	96	9	-131	77	-252	344	130
					C				A					G	

sig2 filtering approach summary:

sig2 basecaller analyzes calibrated peak intensities per channel

```
>SBO8_8_120_6_1512  |  LINE: 692  [ FAIL: 69 ]  ::  LONG__Q: 24 :: CLEAN_LENGTH: 44
 [ ABCD_Q:56  F_QUAL:29 ]  [ Q_FRACT:66 ]
AAAAACAGATGCTACTTACTTTACTACCATATTTCCCAAAACAANGNnCNAACAGnANNAnAnNNCNC*N*NNnnNnnnnNNnnN
```

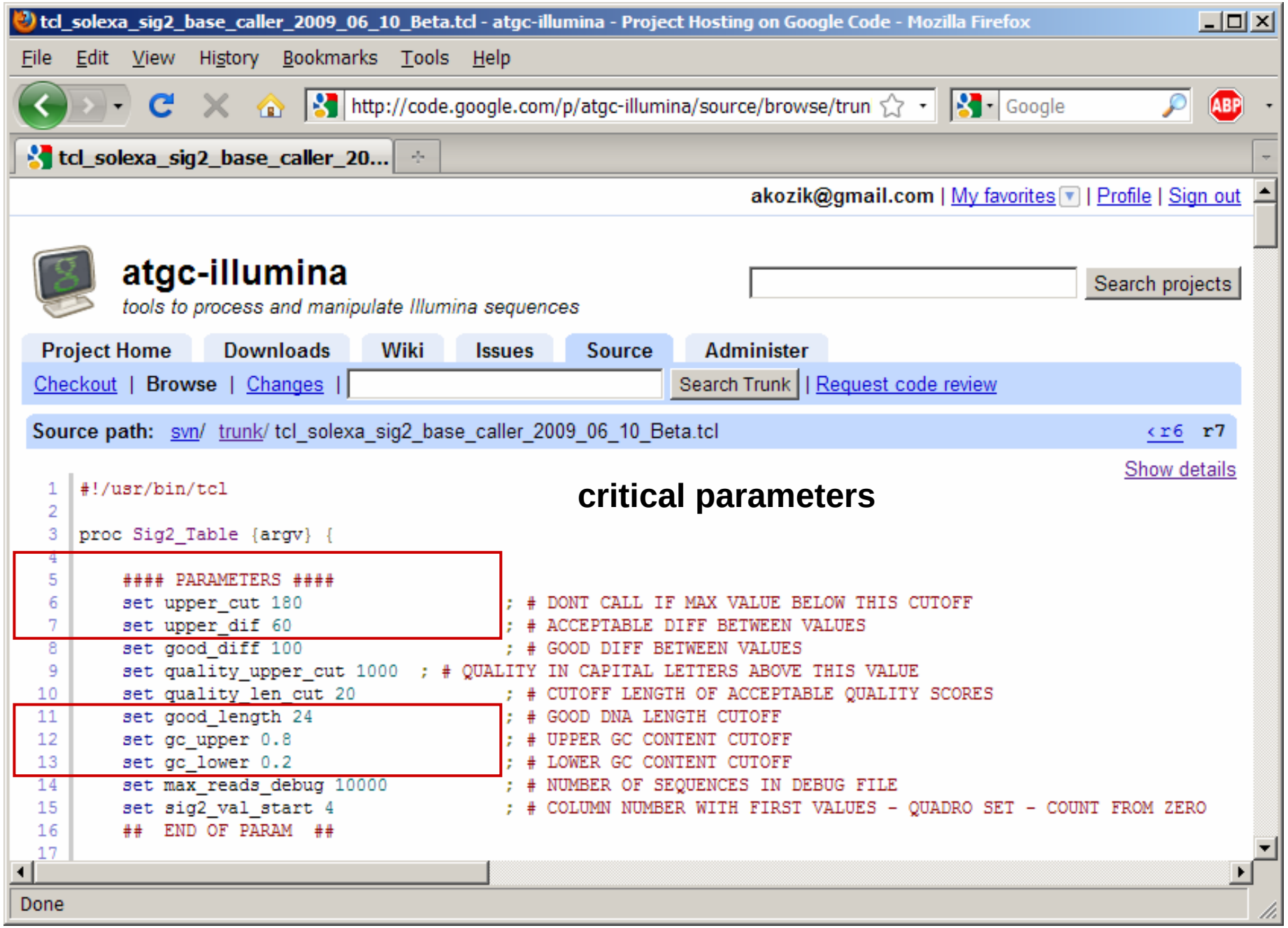
low quality sequence after first failed call is removed

```
>SBO8_8_120_6_1512  |  LINE: 692  [ LENGTH: 44 ]  ( ___GC___: 13/44 )
 [ ABCD_Q:44  F_QUAL:0 ]  [ Q_FRACT:100 ]
AAAAACAGATGCTACTTACTTTACTACCATATTTCCCAAAACAA
```

future plans - to find correlation between IGA quality scores and peak intensities

```
AAAAACAGATGCTACTTACTTTACTACCATATTTCCCAAAACAANGNnCNAACAGnANNAnAnNNCNC*N*NNnnNnnnnNNnnN
a`aab_Z_V__aXaa\[aa``^a`[^]ab`aa^aa_a]SXZZ]`UXXS_Y`__^^S^^Y^S^GU]^]TDIDL]SSZYPWU[[[V[
```

sig2 basecall / filtering



example of basecalling and filtering for sig2 files

	A	C	G	T			basecall					diff		ratio		qual
>SBO8_8_120_5_1105	COUNT: 636															
1:	A:-249	C:44	G:523	T:315	GOOD	BASE_CALL:->	G	FIRST_FAIL:	--	MAX_DIFF:	208	RATIO:	0.60	QUAL:	d
2:	A:-248	C:79	G:25	T:443	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	364	RATIO:	0.17	QUAL:	a
3:	A:-146	C:-138	G:146	T:535	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	389	RATIO:	0.27	QUAL:	b
4:	A:-15	C:-177	G:159	T:372	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	213	RATIO:	0.42	QUAL:	c
5:	A:-241	C:-77	G:79	T:667	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	588	RATIO:	0.11	QUAL:	a
6:	A:2	C:-25	G:-10	T:558	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	556	RATIO:	0.00	QUAL:	a
7:	A:-54	C:-72	G:-1	T:561	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	562	RATIO:	-0.00	QUAL:	a
8:	A:281	C:349	G:-13	T:-55	GOOD	BASE_CALL:->	c	FIRST_FAIL:	--	MAX_DIFF:	68	RATIO:	0.81	QUAL:	f
9:	A:-67	C:-138	G:359	T:226	GOOD	BASE_CALL:->	G	FIRST_FAIL:	--	MAX_DIFF:	133	RATIO:	0.62	QUAL:	d
10:	A:-160	C:-58	G:123	T:440	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	317	RATIO:	0.27	QUAL:	b
11:	A:-260	C:-21	G:6	T:668	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	662	RATIO:	0.00	QUAL:	a
12:	A:-17	C:-55	G:-31	T:613	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	630	RATIO:	-0.02	QUAL:	a
13:	A:250	C:306	G:-35	T:-5	HOPE	BASE_CALL:->	x	FIRST_FAIL:	--	MAX_DIFF:	56	RATIO:	0.81	QUAL:	f
14:	A:176	C:-13	G:-23	T:317	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	141	RATIO:	0.55	QUAL:	c
15:	A:318	C:7	G:32	T:17	GOOD	BASE_CALL:->	A	FIRST_FAIL:	--	MAX_DIFF:	286	RATIO:	0.10	QUAL:	a
16:	A:194	C:303	G:69	T:-20	GOOD	BASE_CALL:->	C	FIRST_FAIL:	--	MAX_DIFF:	109	RATIO:	0.64	QUAL:	d
17:	A:-86	C:13	G:-56	T:461	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	448	RATIO:	0.02	QUAL:	a
18:	A:353	C:-84	G:0	T:157	GOOD	BASE_CALL:->	A	FIRST_FAIL:	--	MAX_DIFF:	196	RATIO:	0.44	QUAL:	c
19:	A:327	C:171	G:-19	T:54	GOOD	BASE_CALL:->	A	FIRST_FAIL:	--	MAX_DIFF:	156	RATIO:	0.52	QUAL:	c
20:	A:149	C:-8	G:-18	T:288	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	139	RATIO:	0.51	QUAL:	c
21:	A:334	C:44	G:46	T:12	GOOD	BASE_CALL:->	A	FIRST_FAIL:	--	MAX_DIFF:	288	RATIO:	0.13	QUAL:	a
22:	A:173	C:62	G:-11	T:218	HOPE	BASE_CALL:->	x	FIRST_FAIL:	--	MAX_DIFF:	45	RATIO:	0.79	QUAL:	d
23:	A:130	C:-238	G:14	T:509	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	379	RATIO:	0.25	QUAL:	b
24:	A:89	C:80	G:-50	T:324	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	235	RATIO:	0.27	QUAL:	b
25:	A:-63	C:-70	G:361	T:74	GOOD	BASE_CALL:->	G	FIRST_FAIL:	--	MAX_DIFF:	287	RATIO:	0.20	QUAL:	b
26:	A:41	C:-6	G:27	T:186	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	145	RATIO:	0.22	QUAL:	b
27:	A:86	C:36	G:-27	T:306	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	220	RATIO:	0.28	QUAL:	b
28:	A:-34	C:40	G:-11	T:346	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	306	RATIO:	0.11	QUAL:	a
29:	A:-147	C:80	G:-5	T:380	GOOD	BASE_CALL:->	T	FIRST_FAIL:	--	MAX_DIFF:	300	RATIO:	0.21	QUAL:	b
30:	A:-101	C:-94	G:389	T:260	GOOD	BASE_CALL:->	G	FIRST_FAIL:	--	MAX_DIFF:	129	RATIO:	0.66	QUAL:	d
31:	A:0	C:0	G:0	T:0	CRAP	BASE_CALL:->	*	FIRST_FAIL:	31	MAX_DIFF:	0	RATIO:	X.XX	QUAL:	x
.....																
59:	A:74	C:54	G:115	T:83	NONE	BASE_CALL:->	n	FIRST_FAIL:	31	MAX_DIFF:	32	RATIO:	0.72	QUAL:	x
60:	A:173	C:-104	G:66	T:38	NONE	BASE_CALL:->	N	FIRST_FAIL:	31	MAX_DIFF:	107	RATIO:	0.38	QUAL:	x

>SBO8_8_120_5_1105 | LINE: 636 [FAIL: 31] :: SHORT_Q 22 ::
CLEAN_LENGTH: 12 [ABCD_Q:56 F_QUAL:29] [Q_FRACT:66]
GTTTTTTcGTTTxTACTAATAxTTGTTTTG*A*AGTGxgCCTCCCTxTGNGGATTCTTnN*GA*Gnn**t*G**nn**nNn*nnn

fasta output

sig2 basecalling and filtering - example of *.good.trim.fasta output file

```
>SBO8_8_120_65_1105  |  LINE: 9728  [ LENGTH: 44 ]  ( ___GC___: 20/44 )
 [ ABCD_Q:44  F_QUAL:0 ]  [ Q_FRACT:100 ]
AACAAATGTCTCGAATCTTAACCGCACATCAAGCTGCAAGAGGG

>SBO8_8_120_65_1326  |  LINE: 9734  [ LENGTH: 46 ]  ( ___GC___: 20/46 )
 [ ABCD_Q:46  F_QUAL:0 ]  [ Q_FRACT:100 ]
AGAACAAGGAGGATTATGCCAAGTTCTACGAGGCTTTCTCCAAGAA

>SBO8_8_120_88_573  |  LINE: 13230  [ LENGTH: 83 ]  ( ___GC___: 22/83 )
 [ ABCD_Q:83  F_QUAL:0 ]  [ Q_FRACT:100 ]
CGAGAATATATATGAACAAGAGTTAGAACAAAAAGAAAGGATGAAAATGAAATCATACAAAATTCTGAATCATTTCTCTAACT

>SBO8_8_120_88_899  |  LINE: 13235  [ LENGTH: 63 ]  ( ___GC___: 17/63 )
 [ ABCD_Q:63  F_QUAL:0 ]  [ Q_FRACT:100 ]
CTCATAAGTAATTCAAAGCTTTTTCAAACTTCATTaCAAATAATTAACGTTTCATTTCCATGG

>SBO8_8_120_166_1586  |  LINE: 25397  [ LENGTH: 48 ]  ( LOWER_GC: 6/48 )
 [ ABCD_Q:48  F_QUAL:0 ]  [ Q_FRACT:100 ]
ATTCTGTTCACTAATGGTTTATTTaTTTATTTATTTTTTTTTTTATAA

>SBO8_8_120_170_1139  |  LINE: 26020  [ LENGTH: 30 ]  ( UPPER_GC: 24/30 )
 [ ABCD_Q:30  F_QUAL:0 ]  [ Q_FRACT:100 ]
CGTCGATCCCCACCACGCCCGGCTCCGCCG
```

GC% content info

grep-able fasta header

sig2 basecalling and filtering of reads with adapter - example of *.good.trim.fastq

```
@SBN4_4_24_177_1635  _TCGTAT_ : 21
TTCAAGGATAACTTTTGCATATCGTAT
+
IIIII99IIIIII9II599III909I9

@SBN4_4_24_178_1771  _TCGTAT_ : 19
TCCCAAATGTAGACAAAGCTCGTATGCCGTCTTCT
+
IIIIIIII99I5IIIII95959II95II599II99

@SBN4_4_24_179_1454  _NOT_FOUND_ : -1
AAATGAACTCAGCCGTAACGCGATTC
+
IIII9IIIIII99I5II995I0I9II

@SBN4_4_24_181_1684  _NOT_FOUND_ : -1
AGTGGTCATGAGAAGAAAGATTGCT
+
I9II9IIII5I5II5III9II95II

@SBN4_4_24_182_1666  _TCGTAT_ : 23
GGGTGAGGGAGATAGATGGATTATCGTAT
+
999I9II59I9III9II95II99I959I9

@SBN4_4_24_183_1747  _TCGTAT_ : 24
AAGAGGGAAAAGGGCTATTAAGCTTCGTAT
+
II9I9IIIIII9999IIIIII5III95II9
```

```
### TRIMMING WITH ADAPTER OPTIONS ###
set number_of_runs $cycle_last
### set trimming_with_adapter "FALSE"
set trimming_with_adapter "TRUE"
set adapter_seqs "TCGTAT"
set adapter_status "_NOT_FOUND_"
### END OF TRIMMING WITH ADAPTER ###
```

comment / uncomment options within source code

```
I-40(73) 9-24(57) 5-20(53) 0-15(48) #-2(35)
```

Phred pseudo-quality scores (ASCII - 33)

grep-able fasta header:

```
grep -A 1 " _TCGTAT_ "

[ to extract all seqs with tag ]

grep -A 1 " _TCGTAT_ : 21"

[ to extract seqs with tag at pos 21 ]
```

illupa rectificator (obsolete?) -

re-format position, intensity, noise raw files into linear form

pos

```
  -0.50  339.55
  -0.47   29.30
  -0.43  364.48
  -0.37  653.60
  -0.37  956.44
....
1793.15 1684.18
1793.15 1756.25
1793.15 1790.75
1793.15 1850.09
1793.15 1911.18
1793.15 1955.35
1793.15 1982.12
1793.18  298.99
1793.46 1776.56
   X       Y
```

int

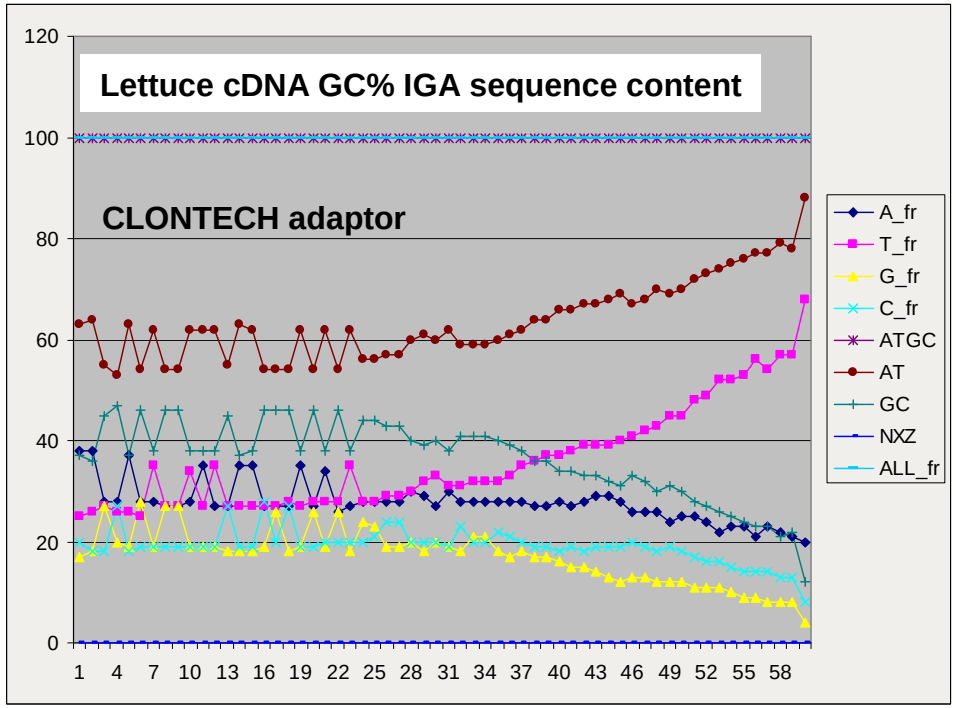
```
 -25.0 -104.9   43.2  163.0
   6.2   28.5 -145.9  502.6
 -12.1  -16.1  -77.0  368.6
   2.9   13.9   12.9  522.6
   7.8   21.2  131.9  215.9
  -1.2    9.3  233.4  342.9
  -2.2   -4.3   -9.6  490.6
   3.6   12.4    5.1  263.3
  -1.8    7.4    2.0   82.2
   2.9  -23.6    4.8  451.0
   6.1   19.9   -2.5  511.6
   5.8   14.7   60.9  140.3
   0.8   10.2  -33.7   39.9
 151.3  584.0 -199.8  -74.3
 188.4  623.9    0.0  -49.1
#END CYCLE 1
```

nse

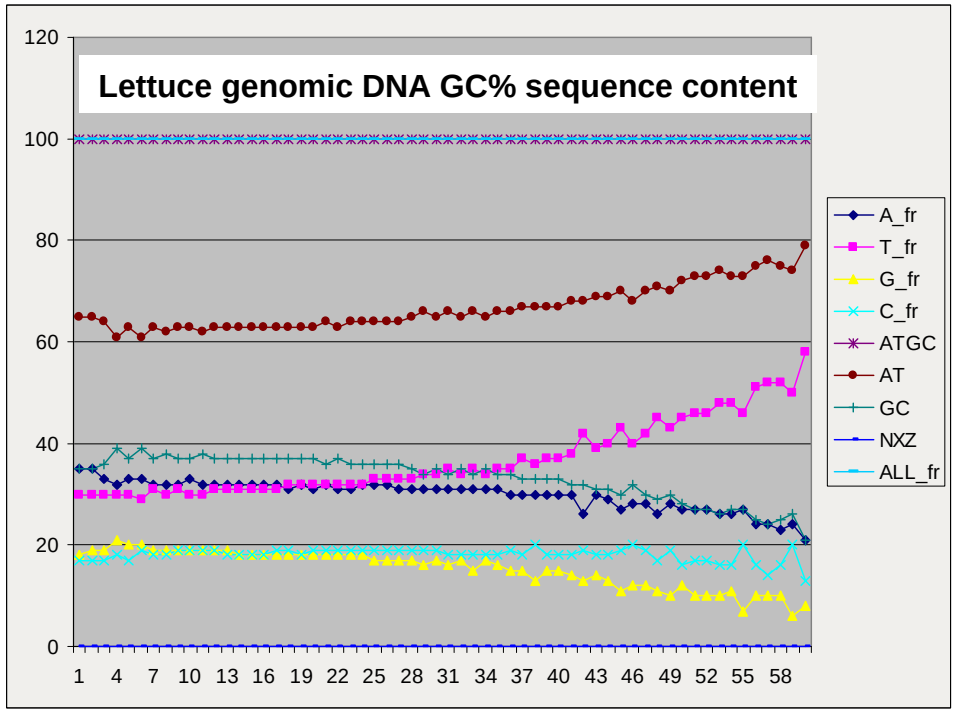
```
   3.1    5.7    2.6    3.3
   3.0    5.8    2.8    5.5
   3.0    5.8    2.8    5.5
   3.0    5.8    2.8    5.5
   3.4    6.3    3.0    3.4
   3.4    6.3    3.0    3.4
   2.8    5.1    2.5    5.3
   2.8    5.1    2.5    5.3
   2.8    5.1    2.5    5.3
   3.2    5.7    2.5    4.4
   3.2    5.7    2.5    4.4
   2.6    5.2    2.4    3.8
   2.6    5.2    2.4    3.8
   3.2    6.4    2.8    4.0
   2.8    5.1    0.0    5.3
#END CYCLE 1
```

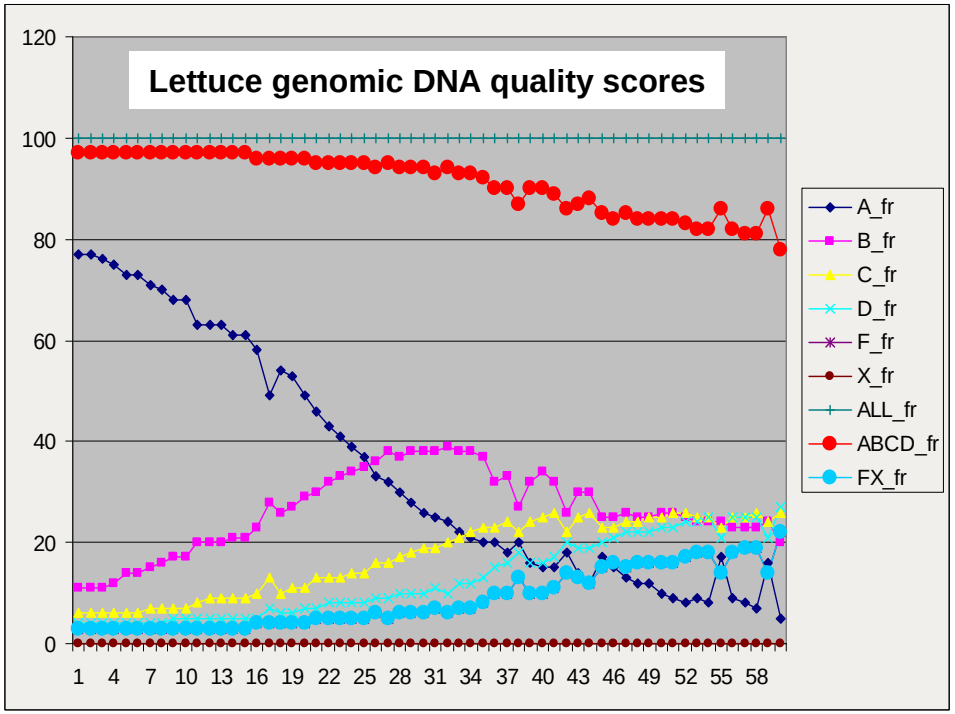
line	tile	X	Y	cycle 1 A C G T	cycle 2 A C G T	cycle 3 A C G T	cycle 4 A C G T
3	0012	108.61	1169.14	-10 20 22 630	9 21 894 479	13 25 756 411	182 539 2 20
3	0012	108.63	1586.81	570 685 1 9	43 37 770 545	38 45 540 394	770 529 -7 15
3	0012	108.67	1305.63	127 503 2 14	11 104 19 670	37 57 703 442	121 13 766 462
3	0012	108.72	1600.48	-98 -105 6 609	-56 528 -32 -18	676 489 -15 -32	651 407 -57 16
3	0012	108.75	1644.07	165 582 2 24	39 54 20 620	208 657 5 -13	868 574 4 -18
3	0012	108.75	1357.78	191 665 1 25	9 26 21 639	182 683 4 24	672 503 13 21
3	0012	108.76	1894.18	-26 -9 445 540	12 12 543 354	794 580 4 17	464 452 5 5
3	0012	108.78	1694.68	175 611 -7 1	-16 -29 18 569	187 633 -3 9	271 581 -16
3	0012	108.82	1556.78	893 519 -4 18	182 545 4 3	15 17 614 343	555 374 8 12
3	0012	108.83	1365.97	148 645 2 27	908 740 3 8	773 635 6 25	694 538 2 24
3	0012	108.88	875.17	10 35 650 362	145 509 2 -43	135 453 0 21	464 364 3 14

QC for GC% content and quality scores

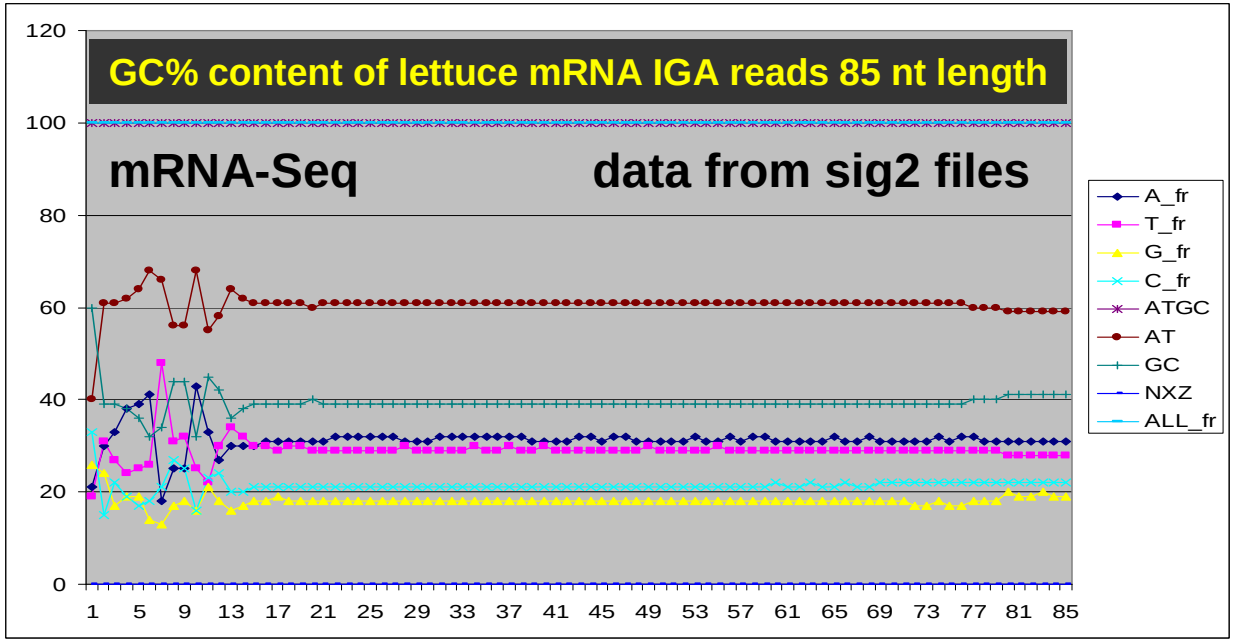


OLD CHEMISTRY

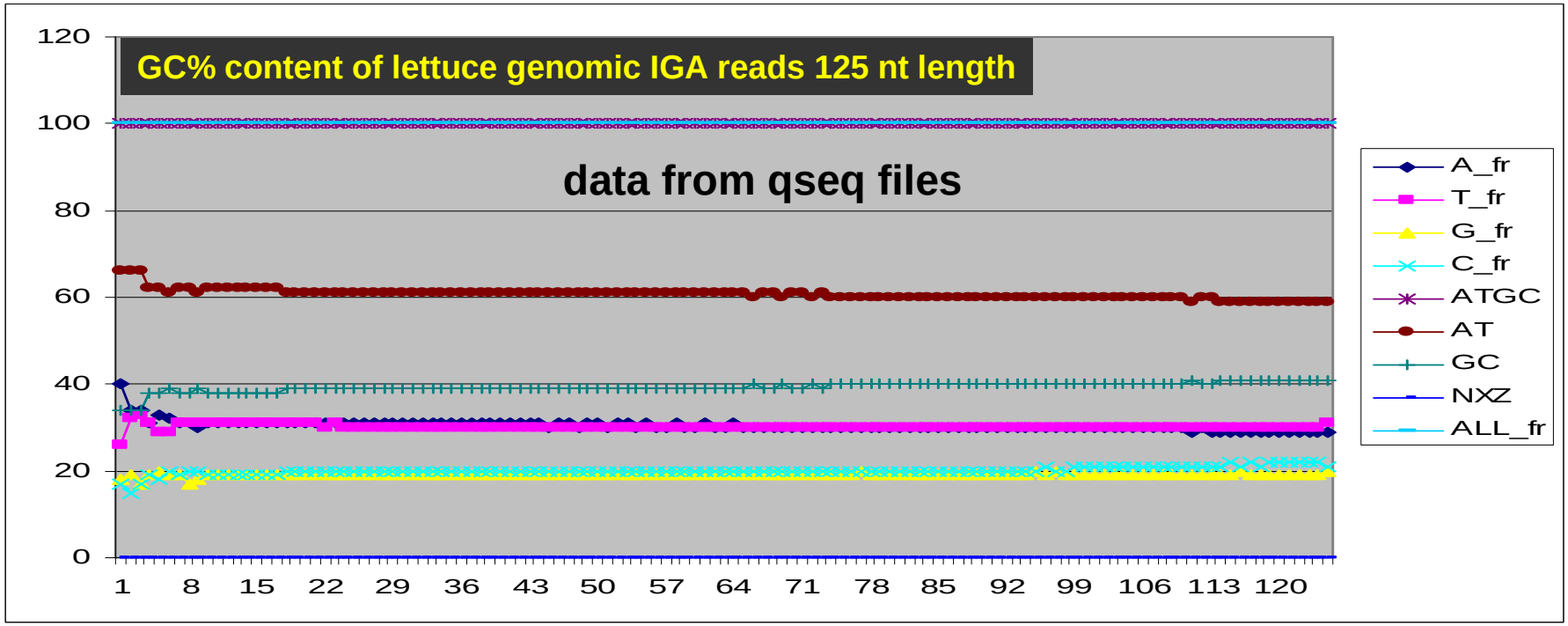


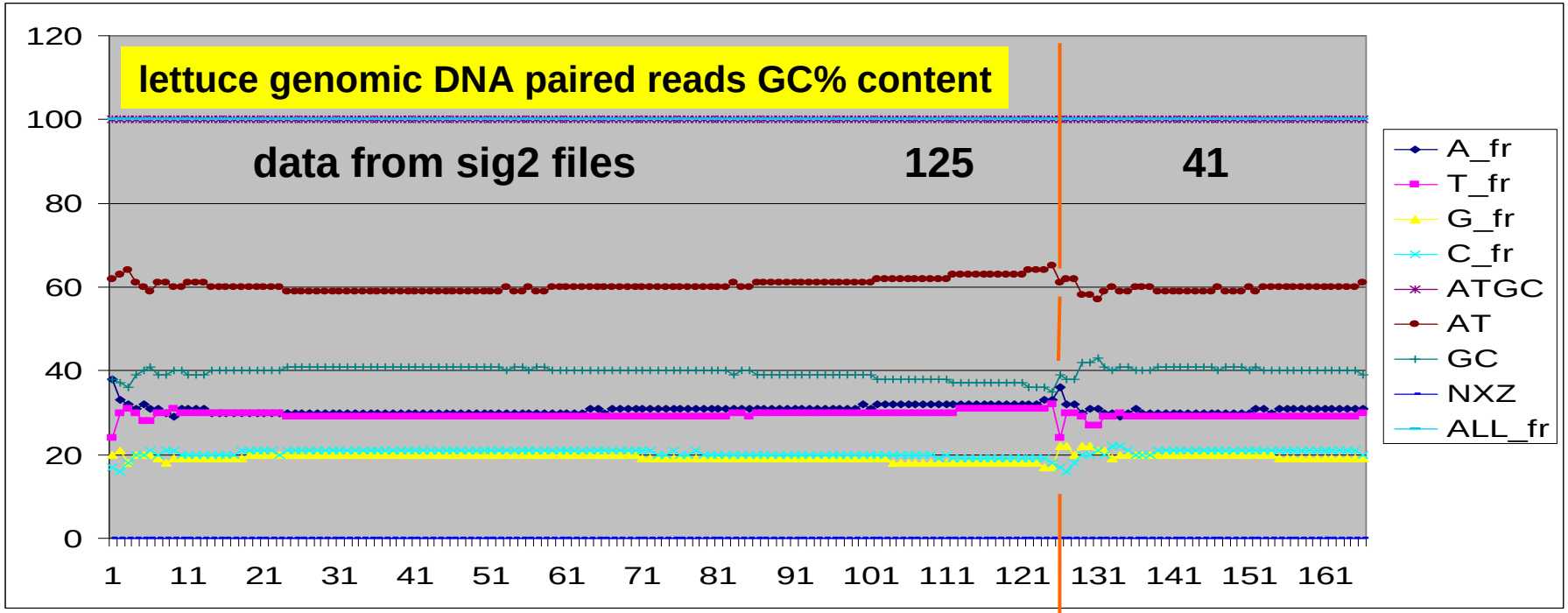


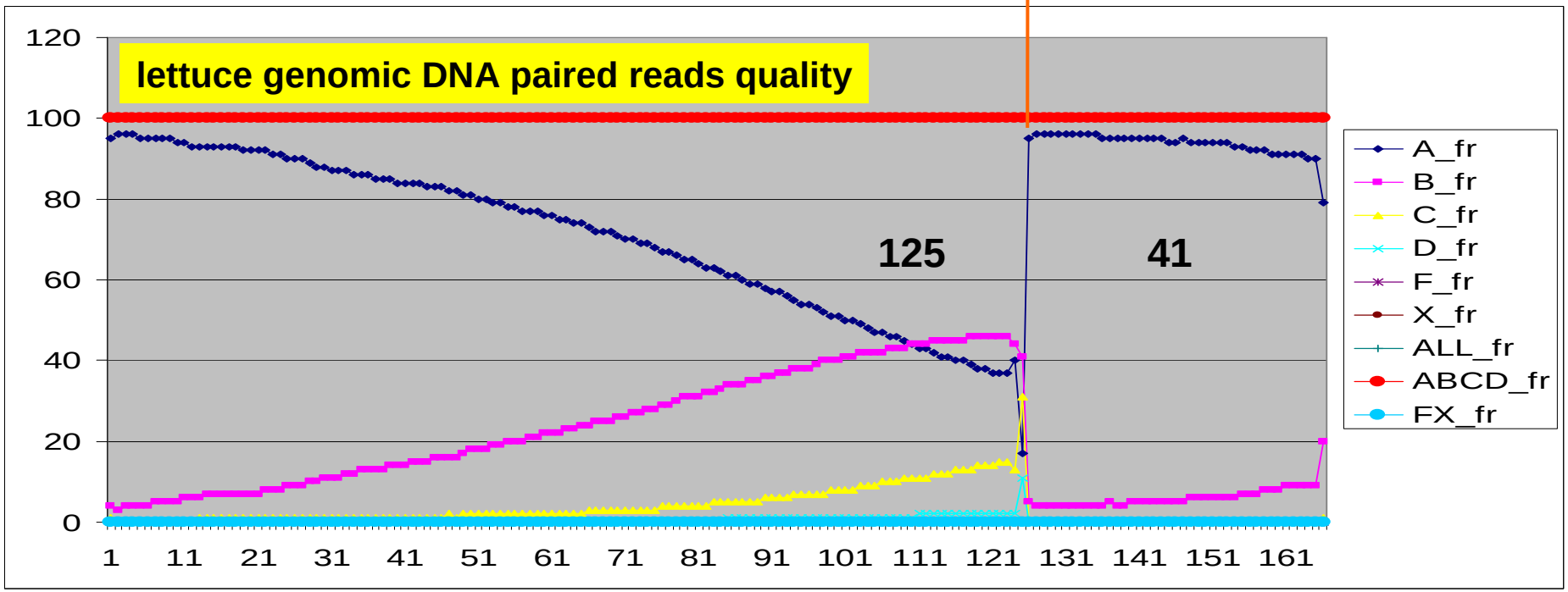
QC examples for GC% content



NEW CHEMISTRY




lettuce genomic DNA paired reads GC% content
data from sig2 files
125
41
120
100
80
60
40
20
0
1
11
21
31
41
51
61
71
81
91
101
111
121
131
141
151
161
A_fr
T_fr
G_fr
C_fr
ATGC
AT
GC
NXZ
ALL_fr


lettuce genomic DNA paired reads quality
125
41
120
100
80
60
40
20
0
1
11
21
31
41
51
61
71
81
91
101
111
121
131
141
151
161
A_fr
B_fr
C_fr
D_fr
F_fr
X_fr
ALL_fr
ABCD_fr
FX_fr

project overview and dataflow

IGA output files

IGA quality scores

filtering for high quality reads

assembly parameters

visualization with Python Contig Viewer

de novo assembly software -

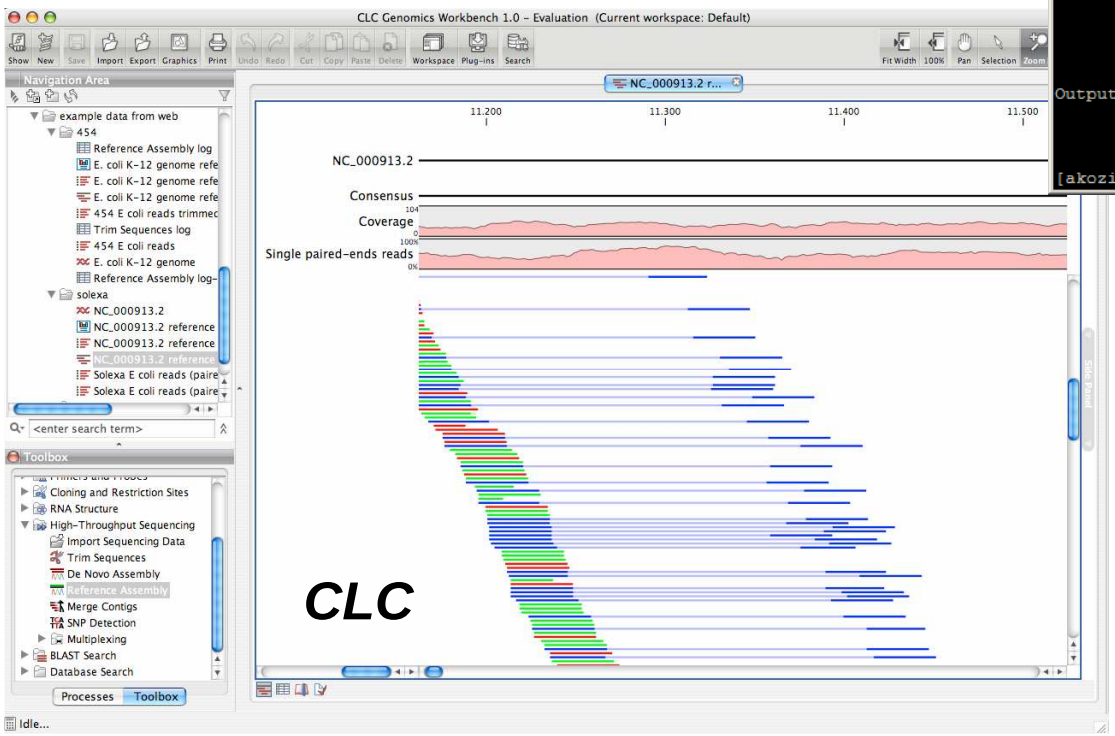
Velvet - publicly available
http://www.ebi.ac.uk/~zerbino/velvet/

CLC Genomics workbench - commercial
http://www.clcbio.com/

different algorithms - different assemblies

```
akozik@aristotle:~/_velvet_0.7.49
[akozik@aristotle _velvet_0.7.49]$ ./velveth
velveth - simple hashing program
Version 0.7.49

Copyright 2007, 2008 Daniel Zerbino (zerbino@ebi.ac.uk)
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compilation settings:
CATEGORIES = 2
MAXKMERLENGTH = 31

Usage:
./velveth directory hash_length {[-file_format][-read_type] filename}

       directory               : directory name for output files
       hash_length             : odd integer (if even, it will be decremented)
<= 31 (if above, will be reduced)
       filename                : path to sequence file or - for standard input

File format options:
       -fasta
       -fastq
       -fasta.gz
       -fastq.gz
       -eland
       -gerald

Read type options:
       -short
       -shortPaired
       -short2
       -shortPaired2
       -long
       -longPaired


Output:
       directory/Roadmaps
       directory/Sequences
              [Both files are picked up by graph, so please leave them there]
[akozik@aristotle _velvet_0.7.49]$
```

Velvet



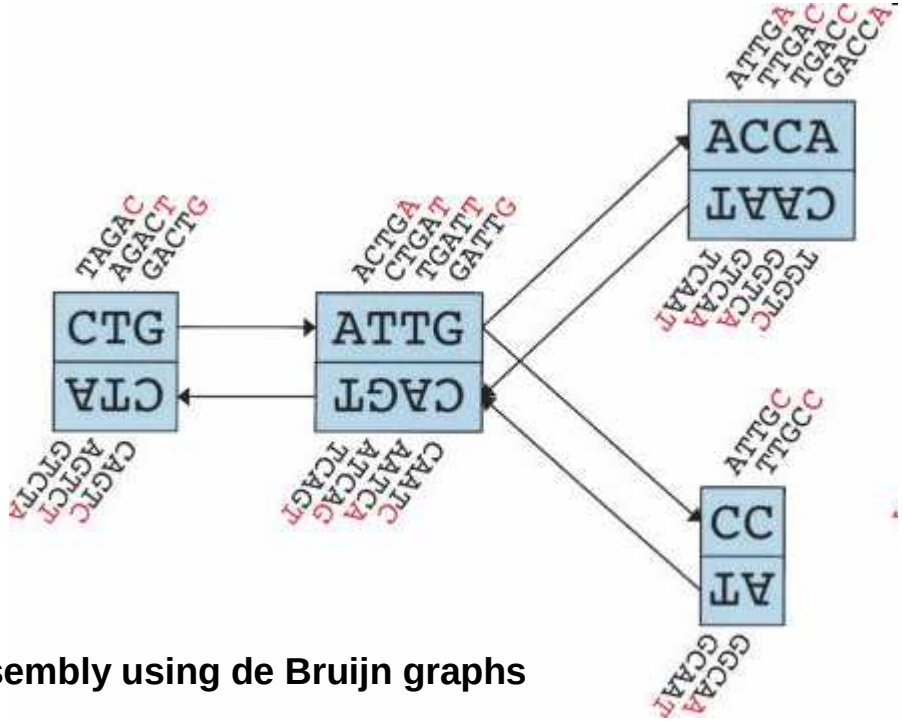
Velvet -
- large input 120+ million reads
- low number of chimeric contigs

CLC -
- 20 million reads input limit
- higher number of chimeras

CLC contigs usually longer than Velvet contigs

Velvet

critical parameter K-mer value



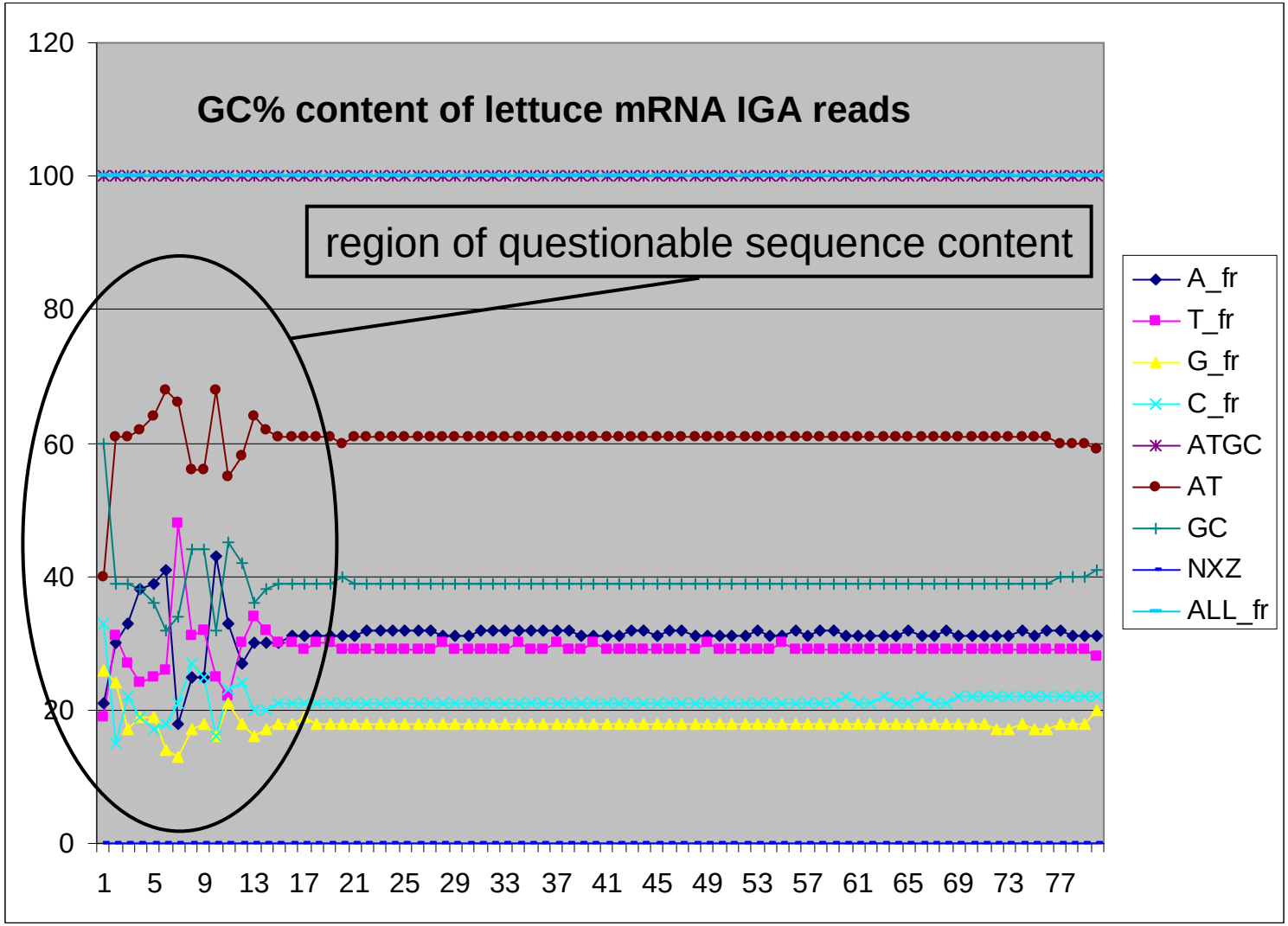
Velvet: Algorithms for de novo short read assembly using de Bruijn graphs
Daniel R. Zerbino and Ewan Birney

Figure 1.
Schematic representation of our implementation of the de Bruijn graph. Each node, represented by a single rectangle, represents a series of overlapping k-mers (in this case, k = 5), listed directly above or below. (Red) The last nucleotide of each k-mer. The sequence of those final nucleotides, copied in large letters in the rectangle, is the sequence of the node. The twin node, directly attached to the node, either below or above, represents the reverse series of reverse complement k-mers. Arcs are represented as arrows between nodes. The last k-mer of an arc's origin overlaps with the first of its destination. Each arc has a symmetric arc. Note that the two nodes on the left could be merged into one without loss of information, because they form a chain.

Genome Res. 2008 May; 18(5): 821–829.
doi: 10.1101/gr.074492.107.

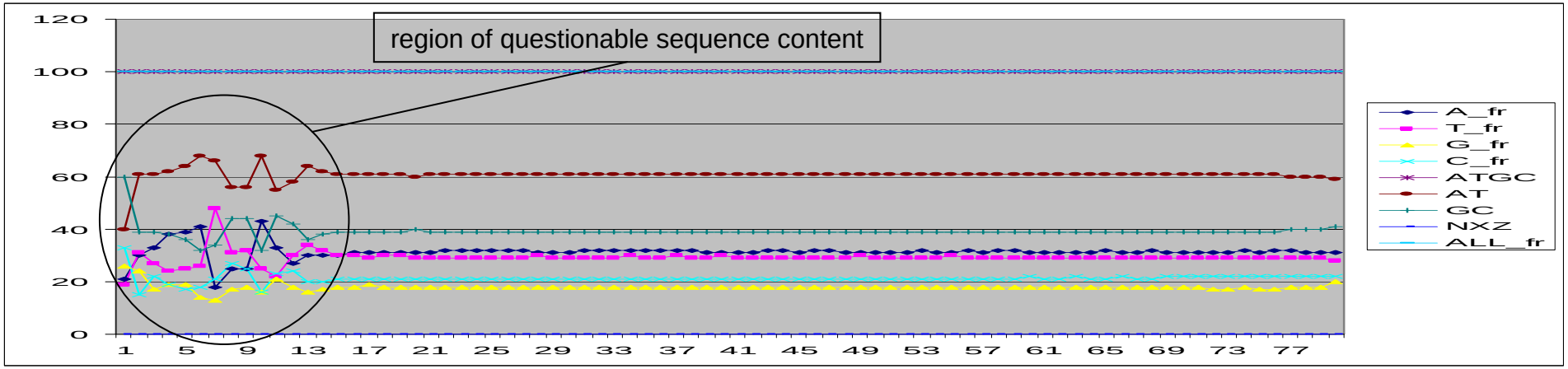
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2336801

case study - assembly of 22.62 million mRNA reads



QC trimming and assembly with 22.62 million reads

four subsets: original 80 nt, and trimmed 70 nt, 65 nt and 60 nt



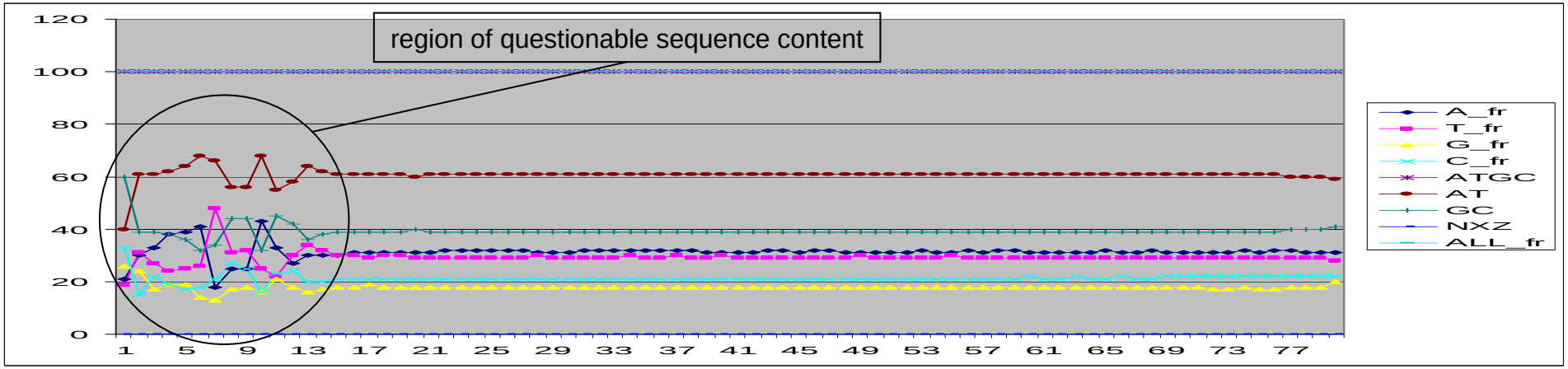

```
$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_80.fa
>SB08_8_120_1651_1715
CAACAAGGCAAACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAAGCAGA

$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_70_D.fa
>SB08_8_120_1651_1715
CAACAAGGCAAACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGG

$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_70_A.fa
>SB08_8_120_1651_1715
AACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAAGCAGA
```

D

A

set ‘A’ - trim left part (5’) of original sequence
set ‘D’ - trim right part (3’) of original sequence

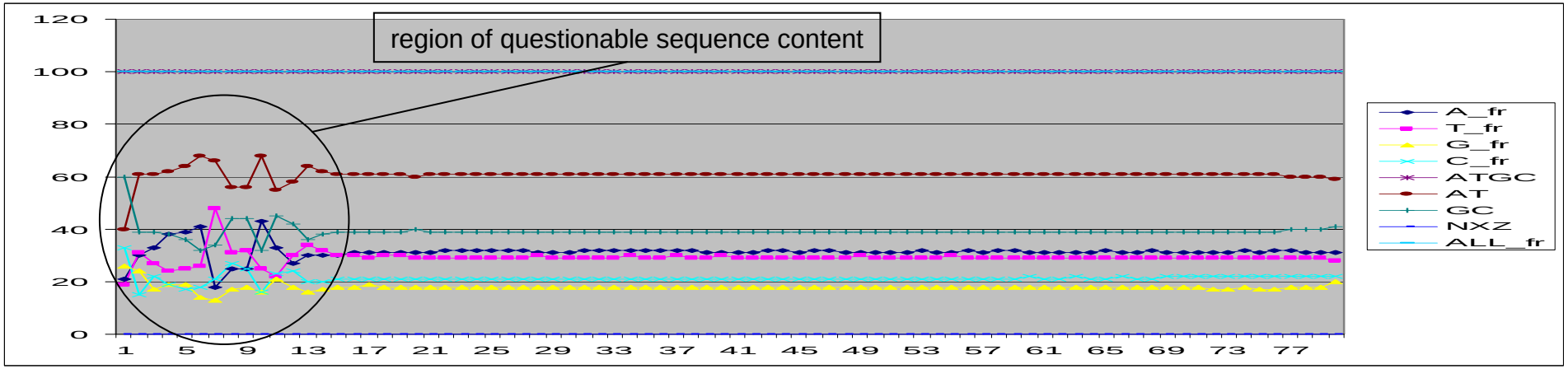
10 nt trimming -> resulting fragments are 70 nt long




```
$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_80.fa
>SBO8_8_120_1651_1715
CAACAAGGCAAACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAAGCAGA

$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_65_D.fa
>SBO8_8_120_1651_1715
CAACAAGGCAAACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGAT
D

$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_65_B.fa
>SBO8_8_120_1651_1715
AACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAA
B

$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_65_A.fa
>SBO8_8_120_1651_1715
TCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAAGCAGA
A
```

set ‘A’ - trim left part (5’) of original sequence
set ‘C’ - trim left and right ends of original sequence
set ‘D’ - trim right part (3’) of original sequence

15 nt trimming -> resulting fragments are 65 nt long




```
bash-2.03$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_80.fa
>SBO8_8_120_1651_1715
CAACAAGGCAAACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAAGCAGA
```

```
$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_60_D.fa
>SBO8_8_120_1651_1715
CAACAAGGCAAACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAA
```

D

```
$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_60_C.fa
>SBO8_8_120_1651_1715
AACCATCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGG
```

C

```
$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_60_B.fa
>SBO8_8_120_1651_1715
TCTTGGACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAA
```

B

```
$ tail -2 090707_SOLEXA1_FC42EK3AAXX_lettuce_cDNA.Clean.GC.Q_60_A.fa
>SBO8_8_120_1651_1715
GACAAAACGTTTGTTGCTTGATGACAAACTATTATGCAAAAAGATCCGGGAAGAAGCAGA
```

A

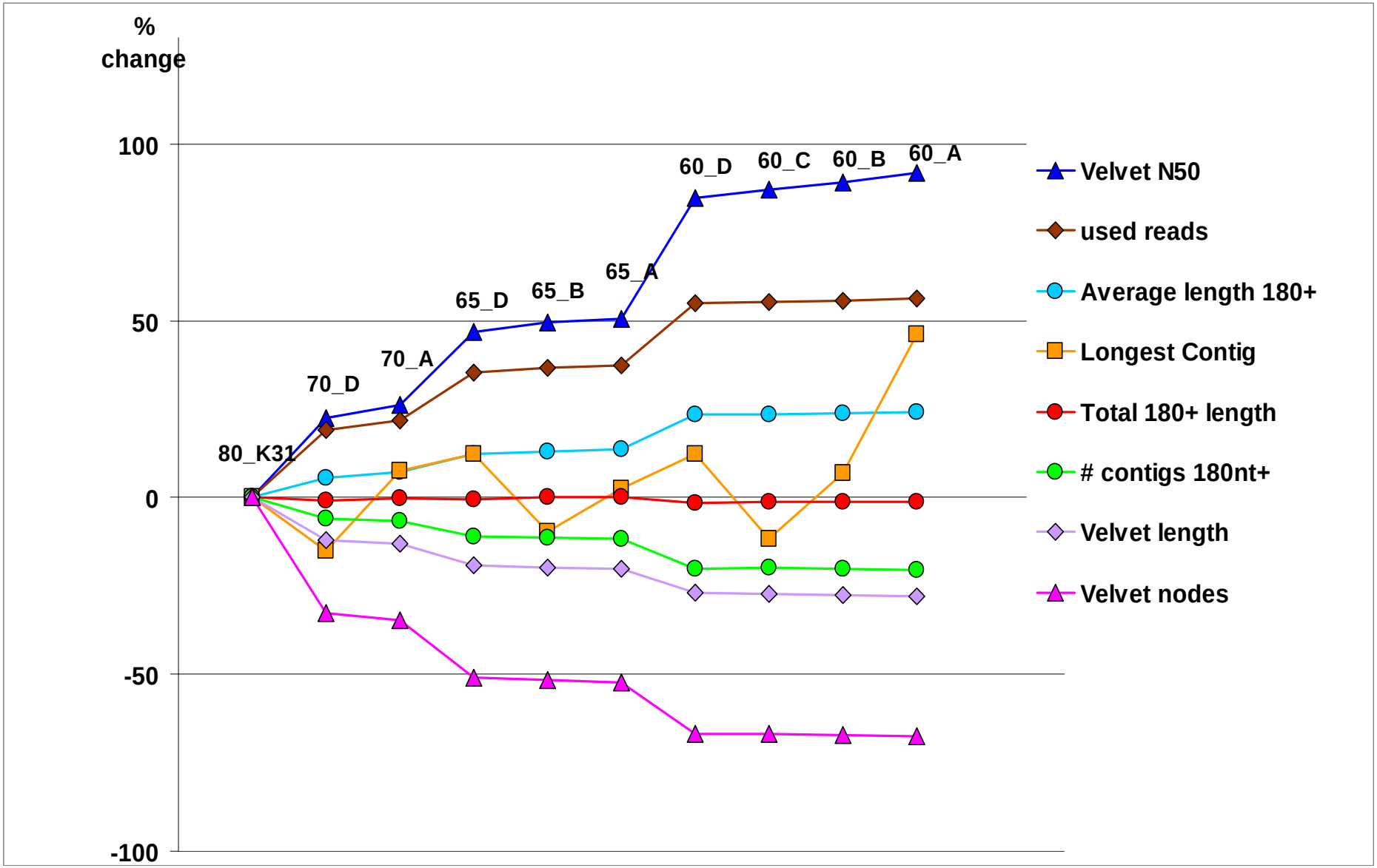
set ‘A’ - trim left part (5’) of original sequence
set ‘B’ and set ‘C’ - trim left and right ends of original sequence
set ‘D’ - trim right part (3’) of original sequence

20 nt trimming -> resulting fragments are 60 nt long

Trimming effect on assemblies (transcriptome): assembly of 22.6 million mRNA-Seq reads of variable length with fixed 31 K-mer value

sample	# of reads (million)	used reads (million)	velvet N50	velvet # nodes	velvet length million nt	longest	median 180 nt +	average 180 nt +	contigs 180 nt +	total 180+ length million nt
80_K27	22.62	8.91	126	968,002	53.08	2428	260	312	79,889	24.94
80_K29	22.62	9.78	141	829,641	50.38	2448	262	318	82,619	26.28
80_K31	22.62	10.73	156	719,028	48.97	4417	267	326	85,123	27.76
70_D	22.62	12.78	191	483,450	43.01	3752	278	344	80,045	27.53
70_A	22.62	**13.08**	**197**	469,949	42.62	**4752**	279	**349**	**79,578**	**27.74**
65_D	22.62	14.53	229	353,683	39.51	4964	290	366	75,602	27.64
65_B	22.62	14.68	233	347,461	39.28	3995	289	368	75,387	27.76
65_A	22.62	**14.75**	**235**	343,477	39.11	**4523**	290	**370**	**75,038**	**27.76**
60_D	22.62	16.61	288	239,076	35.71	4964	307	402	67,942	27.29
60_C	22.62	16.67	292	237,207	35.57	3900	305	403	68,112	27.41
60_B	22.62	16.71	295	235,140	35.44	4720	306	404	67,836	27.41
60_A	22.62	**16.76**	**299**	233,365	35.33	**6459**	306	**405**	**67,744**	**27.43**

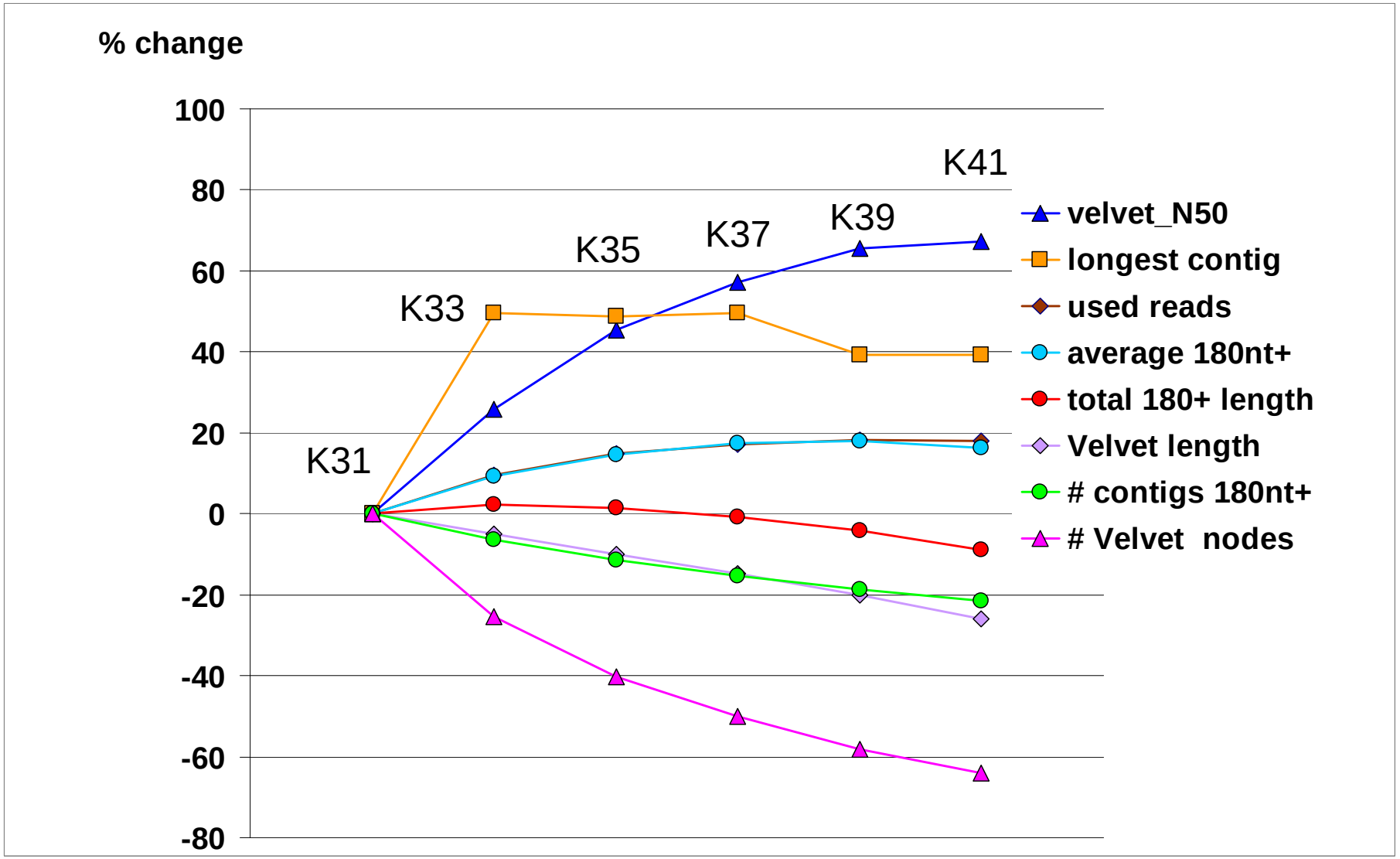
Trimming effect on assemblies (transcriptome): assembly of 22.6 million mRNA-Seq reads of variable length with fixed 31 K-mer value



Velvet K-mer effect on assemblies (transcriptome): assembly of 47.5 million mRNA-Seq reads of fixed 60 nt length with variable K-mer value

sample	# of reads (million)	used reads (million)	velvet N50	velvet # nodes	velvet length million nt	longest	median 180 nt +	average 180 nt +	contigs 180 nt +	total 180+ length million nt
80_QC K31	47.50	17.03	126	1,160,217	60.85	4272	261	318	93,175	29.67
60_QC K31	47.50	33.44	303	325,378	42.83	5131	313	422	76,994	32.55
60_QC K33	47.50	36.64	381	242,908	40.68	7682	328	461	72,092	33.26
60_QC K35	47.50	38.39	441	194,847	38.58	7640	338	484	68,241	33.03
60_QC K37	47.50	39.17	476	162,163	36.46	7682	342	495	65,230	32.26
60_QC K39	**47.50**	**39.58**	**502**	**136,282**	**34.20**	**7141**	**343**	**498**	**62,557**	**31.16**
60_QC K41	47.50	39.47	507	117,446	31.76	7141	338	491	60,446	29.68

Velvet K-mer effect on assemblies (transcriptome): assembly of 47.5 million mRNA-Seq reads of fixed 60 nt length with variable K-mer value



Transcriptome assembly - large scale project - 77.5 million reads

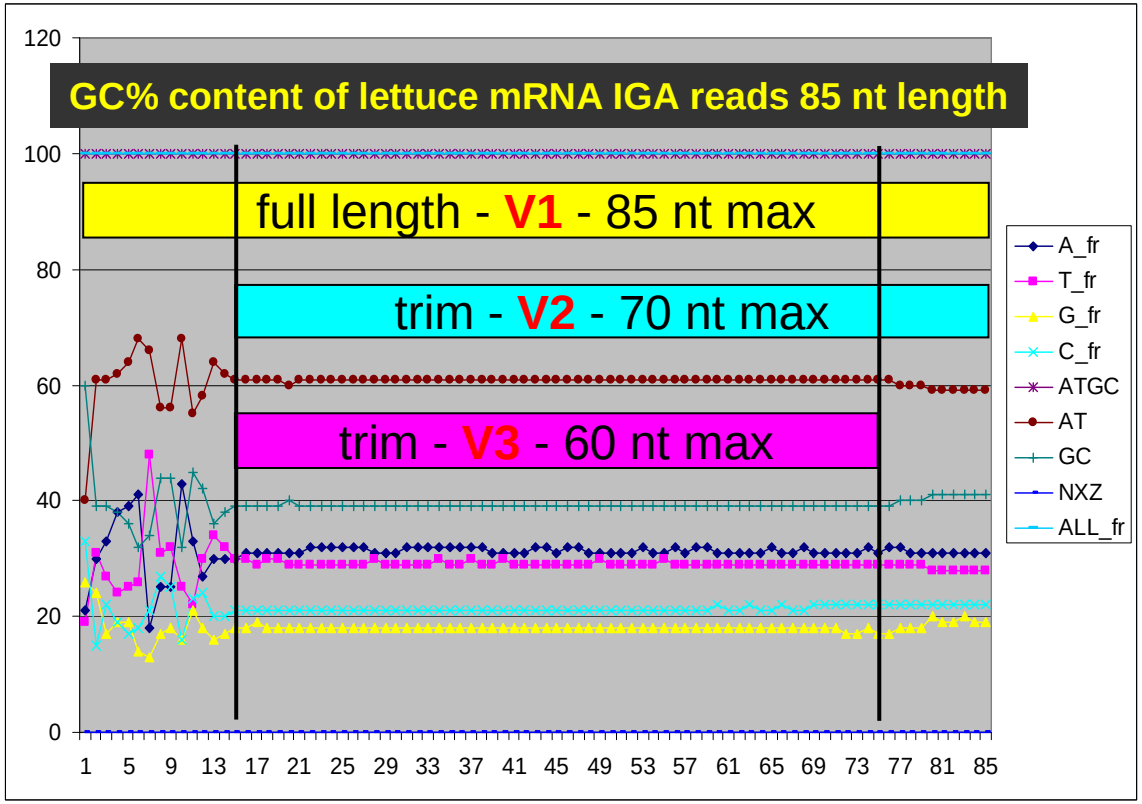
assembly of the assemblies - so called ‘superassembly’

1 step

Several rounds of Velvet assemblies with different parameters and sampling

2 step

CAP3 assembly of different subsets of Velvet contigs - superassembly



3 step

Visualization and validation

3 subsets with trimming options:

V1 - reads within range 40 - 85
(full filtered length ‘as is’)

V2 - reads within range 25 - 70
(trim first 15 nt)

V3 - reads within range 25 - 60
(trim first 15 and last 10 nt)

Transcriptome assembly - large scale project - 77.5 million reads

Velvet

sample	# of reads (million)	used reads (million)	velvet N50	velvet # nodes	velvet length million nt	longest	median 300 nt +	average 300 nt +	contigs 300 nt +	total 300+ length million nt
V1_K35	77.5	15.8	138	1,097,943	63.8	3745	413	489	41,321	20.2
V1_K41(*)	77.5	31.5	251	514,297	54.3	6673	453	562	51,068	28.7
V1_K47	77.5	47.5	482	230,756	47.1	8711	522	691	50,186	34.7
V1_K51	77.5	49.6	580	166,477	43.3	10258	544	729	47,486	34.6
V1_K55	60.9 [1]	46.8	653	124,677	39.6	10843	557	749	44,852	33.6
V2_K37	77.5	39.4	367	290,027	45.4	6547	496	634	45,608	28.9
V2_K41(*)	77.5	45.0	496	186,892	41.2	10139	532	697	43,208	30.1
V2_K45	77.5	45.5	568	133,766	37.1	7170	547	719	40,457	29.1
V2_K49	60.9 [2]	43.4	589	100,648	32.6	7141	542	709	37,677	26.7
V3_K33	60.9 [3]	35.8	361	292,805	44.1	8441	499	640	42,805	27.4
V3_K39	60.9 [3]	44.8	536	154,636	37.6	10394	548	717	39,041	28.0
V3_K43	60.9 [3]	46.0	565	112,506	32.8	7141	546	712	35,948	25.6

Notes: (*) - identical K-mer; different trimming; [1], [2], [3] - subsets of short reads excluded from Velvet input

CAP3 assembly of different subsets of Velvet contigs - superassembly

Input for super assembly:

519,657 contigs 300 nt or longer after 12 iterations of Velvet assembly

Total number of non-redundant Velvet contigs in 12 subsets - 428,122

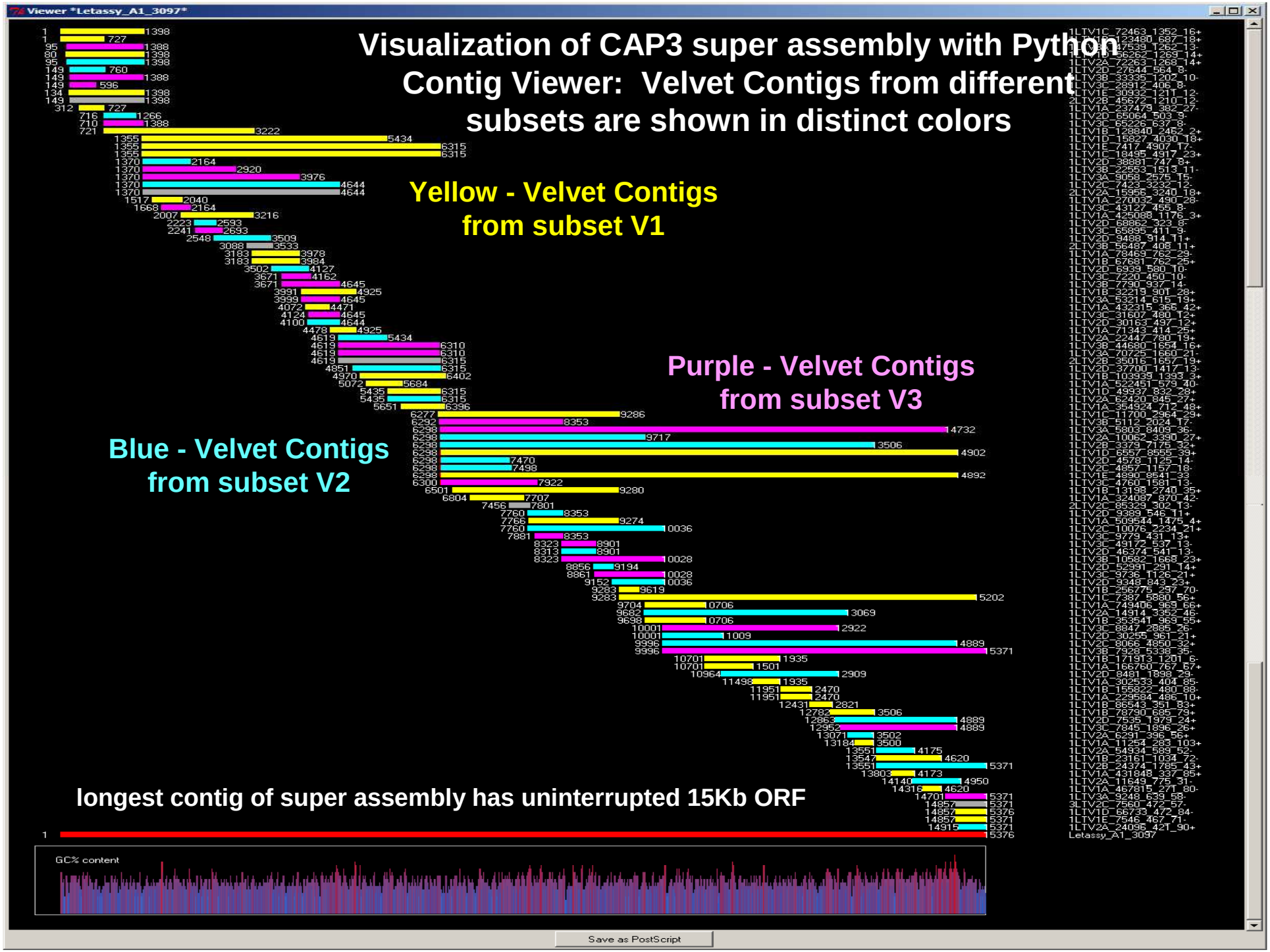
CAP3 assembly with non-redundant set
(parameters: 80nt overlap, 95% identity)

Super assembly output - 58,142 unigenes:
40,853 new CAP3 contigs and 17,289 non-assembled Velvet contigs

longest contig: 15,376 nt - uninterrupted 15 Kb ORF
total length of super assembly: 52,397,565 nt
average contig length: 901 nt
median: 685 nt

BLASTX analysis - super assembly versus Plant Reference Database
1e-10 expect cutoff: 42,633 contigs (73%) are with hits to known proteins

16,446 long contigs (1,000nt+) out of 17,000 (97%) have hits to known protein sequences


Viewer *Letassy_A1_3097*
Visualization of CAP3 super assembly with Python Contig Viewer: Velvet Contigs from different subsets are shown in distinct colors
Yellow - Velvet Contigs from subset V1
Blue - Velvet Contigs from subset V2
Purple - Velvet Contigs from subset V3
longest contig of super assembly has uninterrupted 15Kb ORF
GC% content
Save as PostScript

Validation of superassembly (Velvet + CAP3)
by BLAST search versus Plant Reference Database

Query ID (assembly contig ID)	Subject (BLAST hit) ID	BLAST hit description (annotation)	identity (%)	perfect matches	alignment length (aa)	translation frame	query start (nt)	query end (nt)	subject start (aa)	subject end (aa)	query length (nt)	subject length (aa)
Letassy_A1_3097	Arab_thal.NP_186875	(NP_186875.2 GI:30678519) { BIG (BIG); binding / ubiquitin-protein ligase/ zinc ion binding } [Arabidopsis thaliana]	63	3255	5159	3	204	15194	1	5096	15376	5098
Letassy_A1_290	Viti_vini.XP_002274451	(XP_002274451.1 GI:225432116) { PREDICTED: similar to vacuolar protein sorting 13C protein-like } [Vitis vinifera]	63	2731	4324	-3	12718	65	42	4285	12879	4286
Letassy_A1_7087	Arab_thal.NP_197702	(NP_197702.1 GI:15237223) { zinc finger (C3HC4-type RING finger) family protein } [Arabidopsis thaliana]	57	2448	4264	1	1	12573	356	4597	12573	4706
Letassy_A1_11111	Viti_vini.XP_002268896	(XP_002268896.1 GI:225461945) { PREDICTED: similar to E3 ubiquitin-protein ligase UPL1 } [Vitis vinifera]	69	2683	3855	-2	11129	45	1	3754	11478	3754
Letassy_A1_38867	Viti_vini.XP_002283711	(XP_002283711.1 GI:225459044) { PREDICTED: hypothetical protein } [Vitis vinifera]	73	2727	3730	2	284	11092	3	3653	11127	3653
Letassy_A1_94	Popu_tric.XP_002327756	(XP_002327756.1 GI:224134102) { predicted protein } [Populus trichocarpa]	81	2826	3466	1	1	10248	459	3881	10451	3881
Letassy_A1_37086	Viti_vini.XP_002274761	(XP_002274761.1 GI:225437162) { PREDICTED: hypothetical protein } [Vitis vinifera]	71	2414	3390	-3	9949	2	1	3325	10362	4546
Letassy_A1_4189	Viti_vini.XP_002283711	(XP_002283711.1 GI:225459044) { PREDICTED: hypothetical protein } [Vitis vinifera]	65	2443	3734	-3	10215	151	3	3653	10262	3653

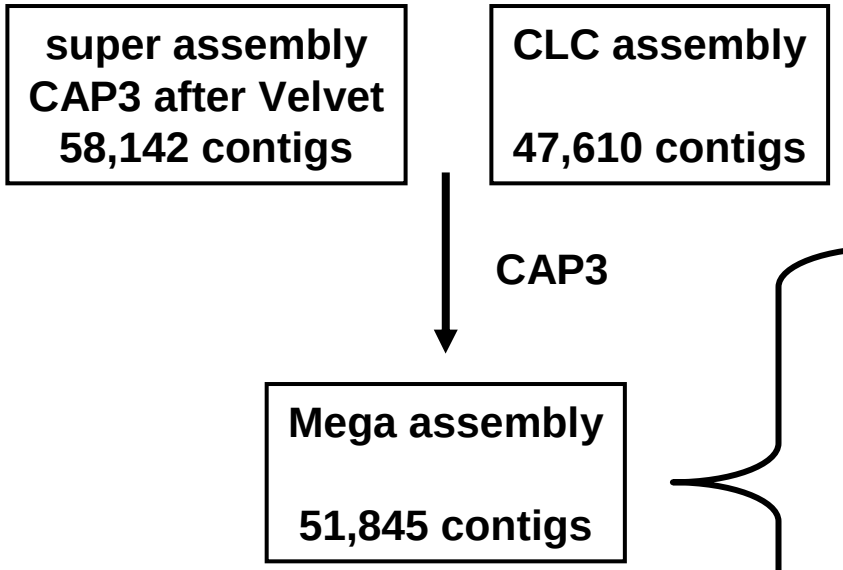
Velvet assembly versus CLC assembly

sample	# of reads (million)	used reads (million)	velvet N50	velvet # nodes	velvet length million nt	longest	median 300 nt +	average 300 nt +	contigs 300 nt +	total 300+ length million nt
V1_K35	77.5	15.8	138	1,097,943	63.8	3745	413	489	41,321	20.2
V1_K41(*)	77.5	31.5	251	514,297	54.3	6673	453	562	51,068	28.7
V1_K47	77.5	47.5	482	230,756	47.1	8711	522	691	50,186	34.7
V1_K51	77.5	49.6	580	166,477	43.3	10258	544	729	47,486	34.6
V1_K55	60.9 [1]	46.8	653	124,677	39.6	10843	557	749	44,852	33.6

CLC ASSEMBLY

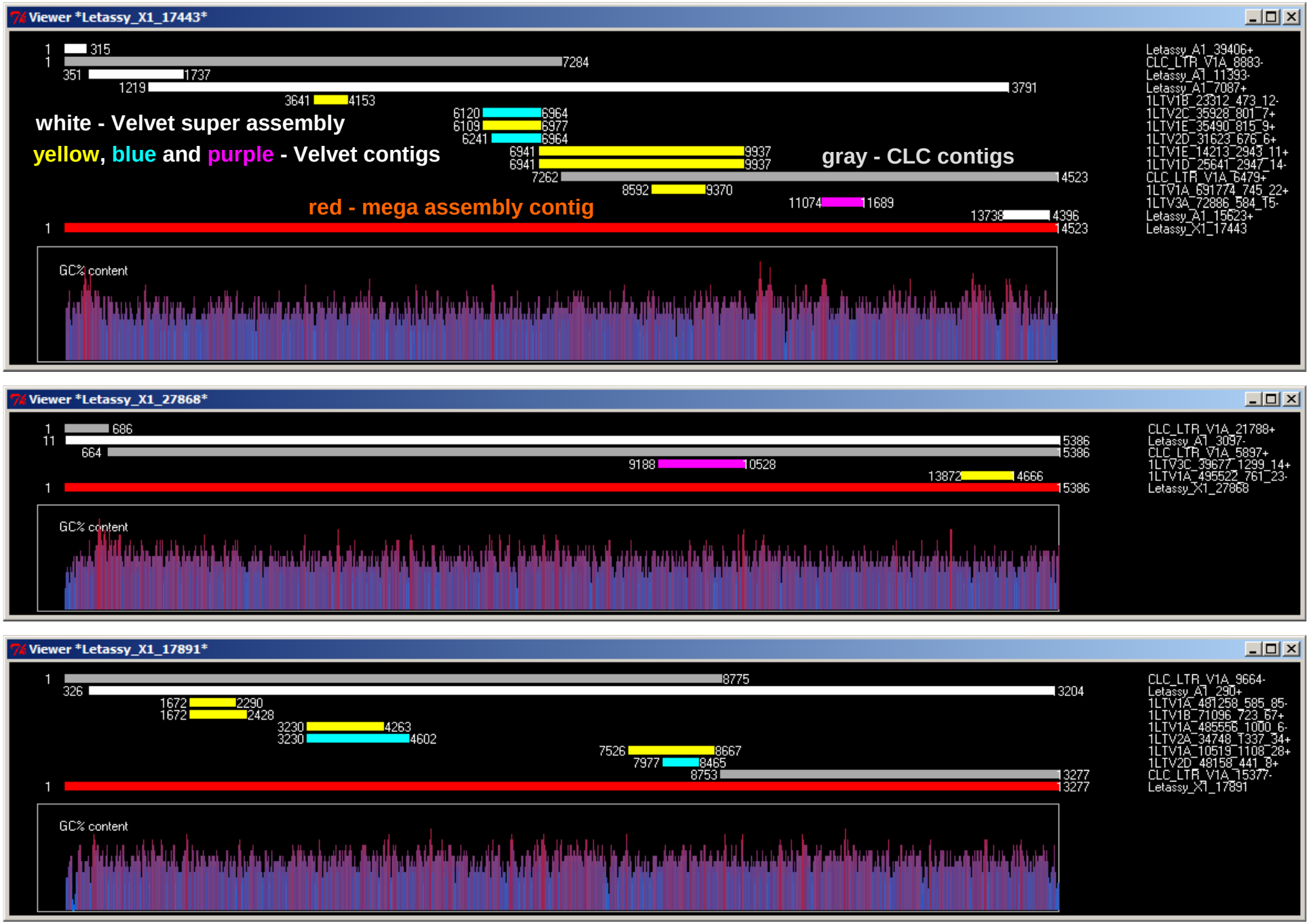
sample	# of reads (million)	used reads (million)				longest	median 300 nt +	average 300 nt +	contigs 300 nt +	total 300+ length million nt
	77.5	**74.5**				**14723**	**697**	**977**	**47,610**	**46.5**

TRANSCRIPTOME MEGA ASSEMLY



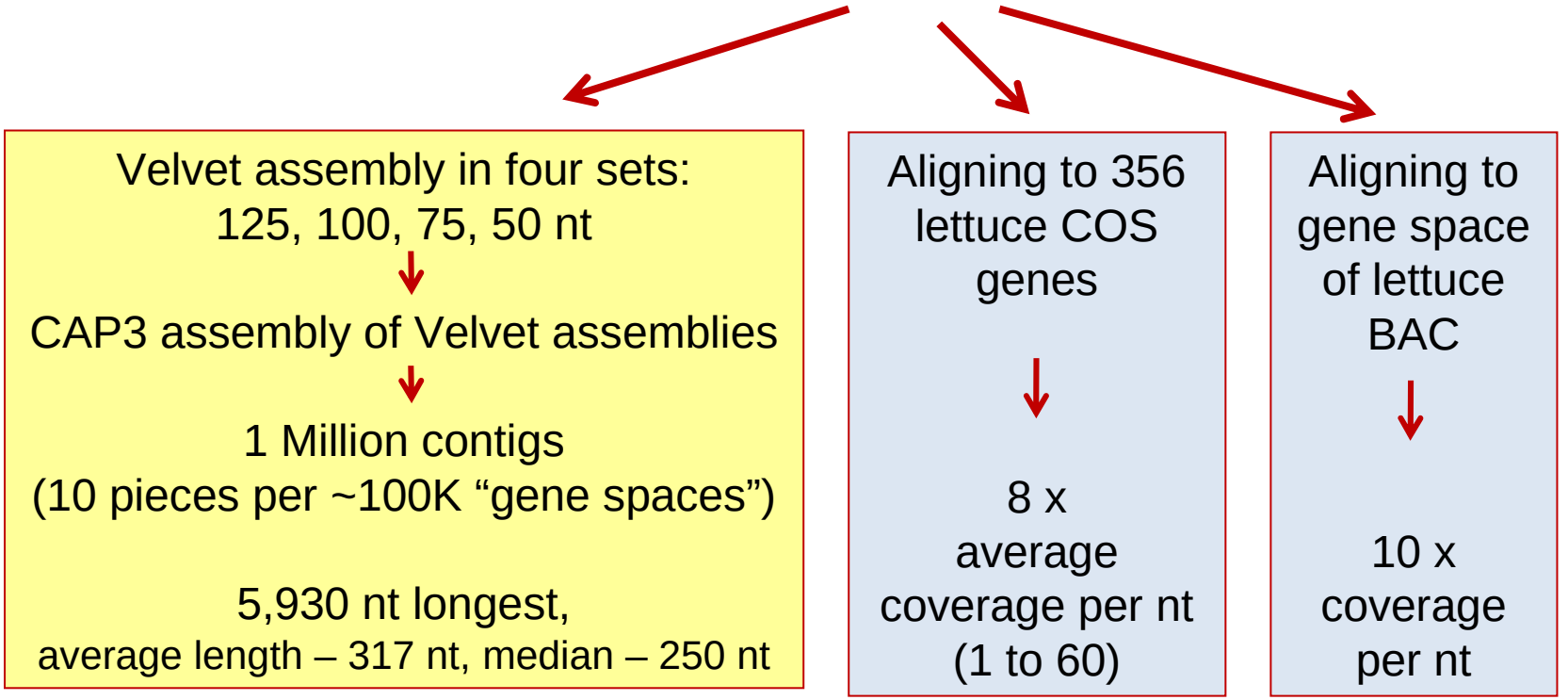
	longest	median 300 nt +	average 300 nt +	contigs 300 nt +	total 300+ length million nt
Velvet/CAP3 12X super assembly	15,376	685	901	58,142	52.4
CLC assembly	14,723	697	977	47,610	46.5
MEGA assembly	15,386	736	1,020	51,845	52.9

MEGA Assembly Visualization with Python Contig Viewer

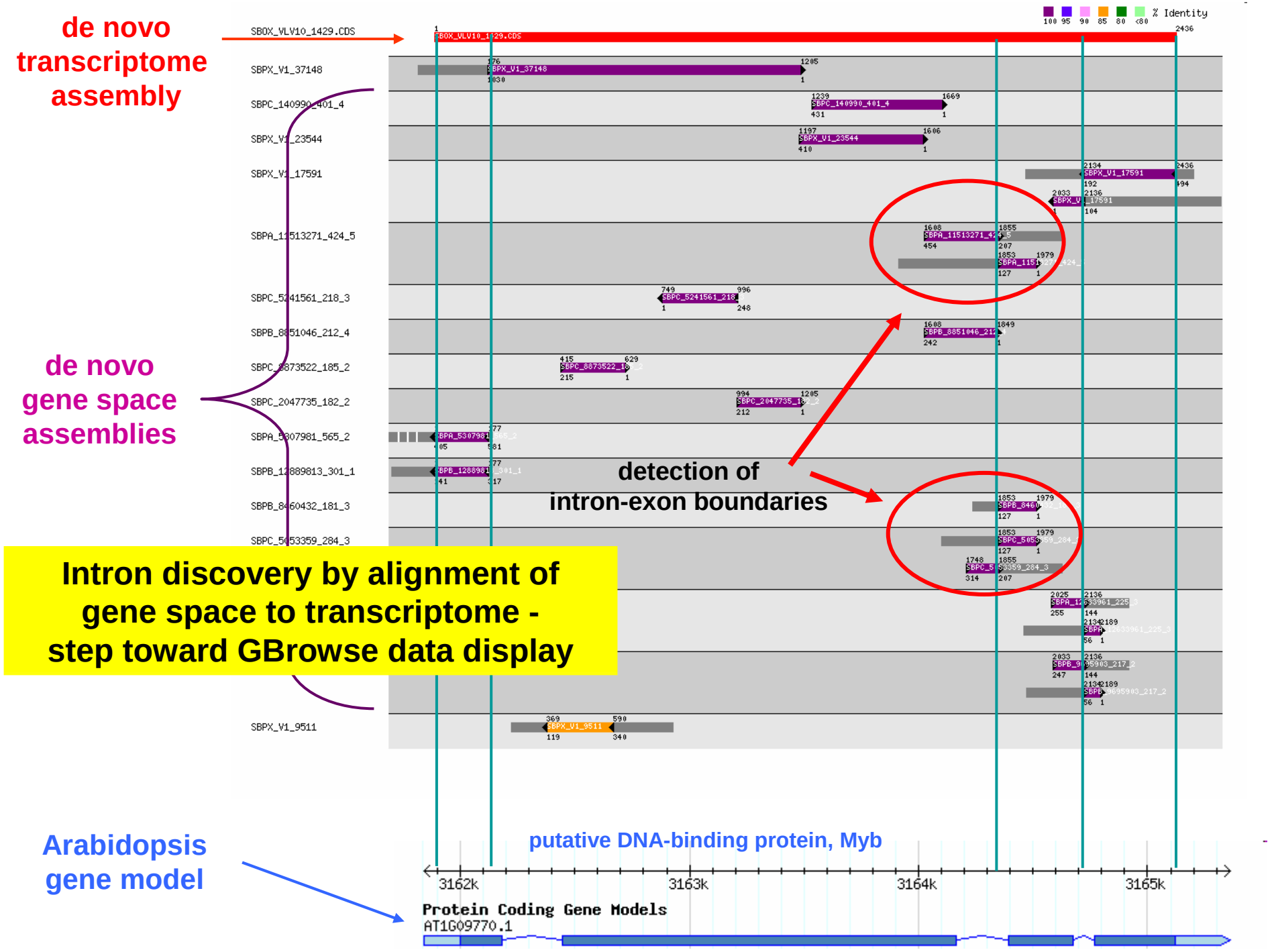


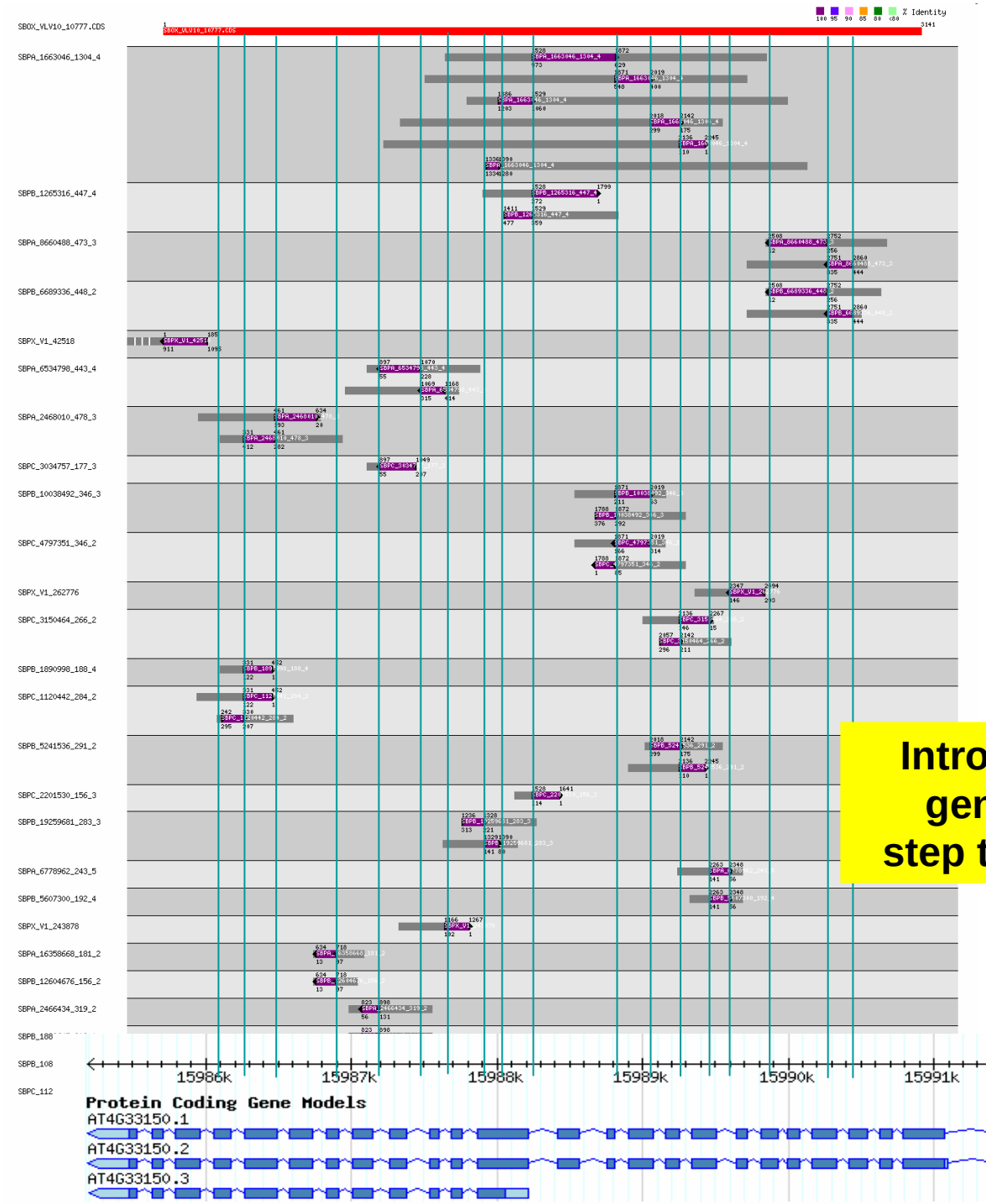
GENOME (GENE SPACE ASSEMBLY)

55 million, 125 nt single reads
6 GB (= 2x coverage w/o repeat reduction)



new data for assembly of ~100 million reads will be available soon (March 2009)


de novo transcriptome assembly
de novo gene space assemblies
Intron discovery by alignment of gene space to transcriptome - step toward GBrowse data display
detection of intron-exon boundaries
Arabidopsis gene model
putative DNA-binding protein, Myb
3162k
3163k
3164k
3165k
Protein Coding Gene Models
AT1G09770.1
% Identity
100 95 90 85 80 <80
SBOX_VLV10_1429.CDS
SBPX_V1_37148
SBPC_140990_401_4
SBPX_V1_23544
SBPX_V1_17591
SBPA_11513271_424_5
SBPC_5241561_218_3
SBPB_8851046_212_4
SBPC_8873522_185_2
SBPC_2047735_182_2
SBPA_5307981_565_2
SBPB_12889813_301_1
SBPB_8460432_181_3
SBPC_5053359_284_3
SBPX_V1_9511



de novo transcriptome assembly

de novo gene space assemblies

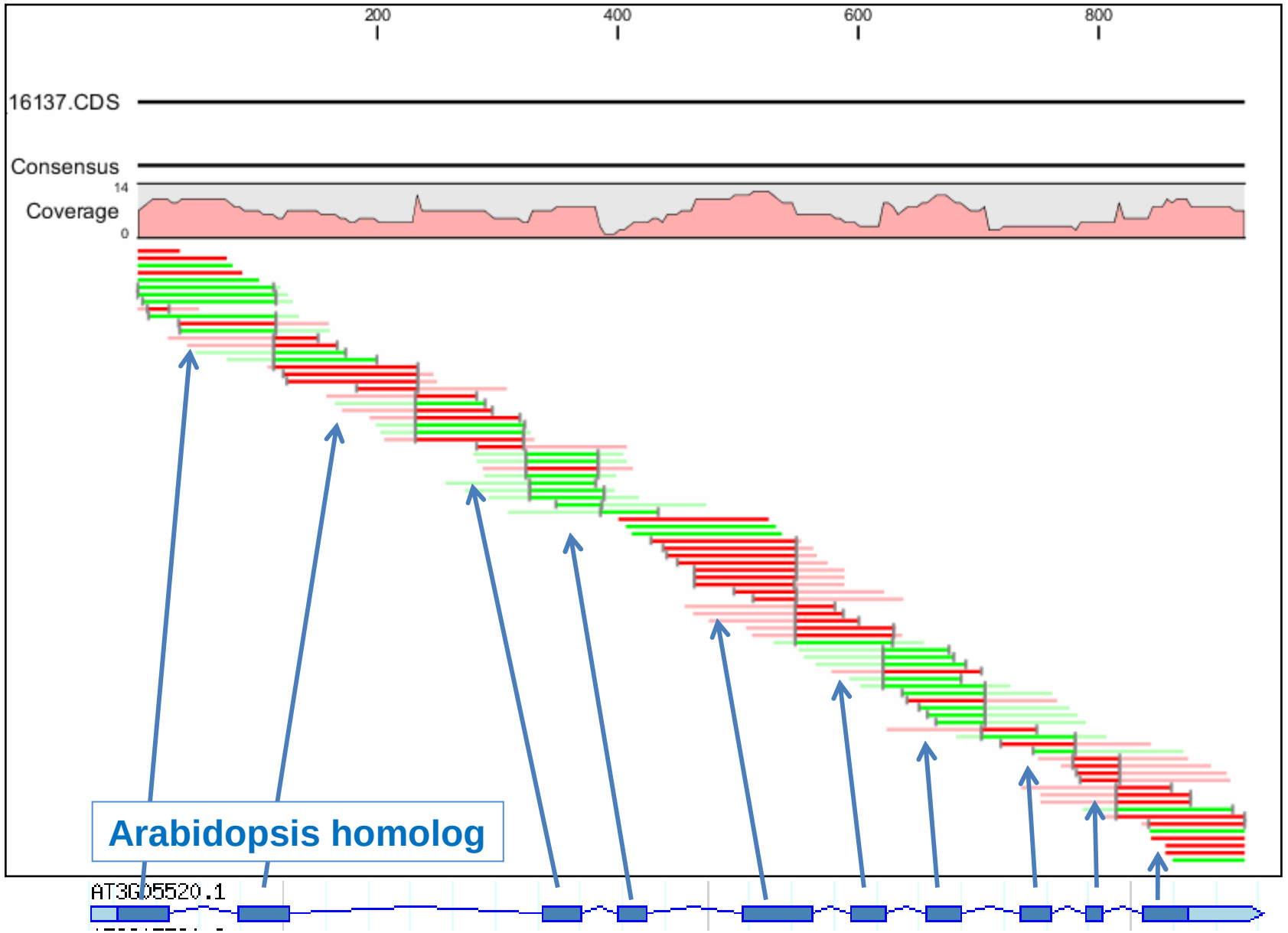
Intron discovery by alignment of gene space to transcriptome - step toward GBrowse data display

lysine-ketoglutarate reductase/saccharopine dehydrogenase (LKR/SDH)

Arabidopsis gene model

How Low Can We Go to Figure the Exon-intron Structure?

E.g. gene covered by only 6.6 genomic reads per nt of coding sequence



project overview and dataflow

IGA output files

IGA quality scores

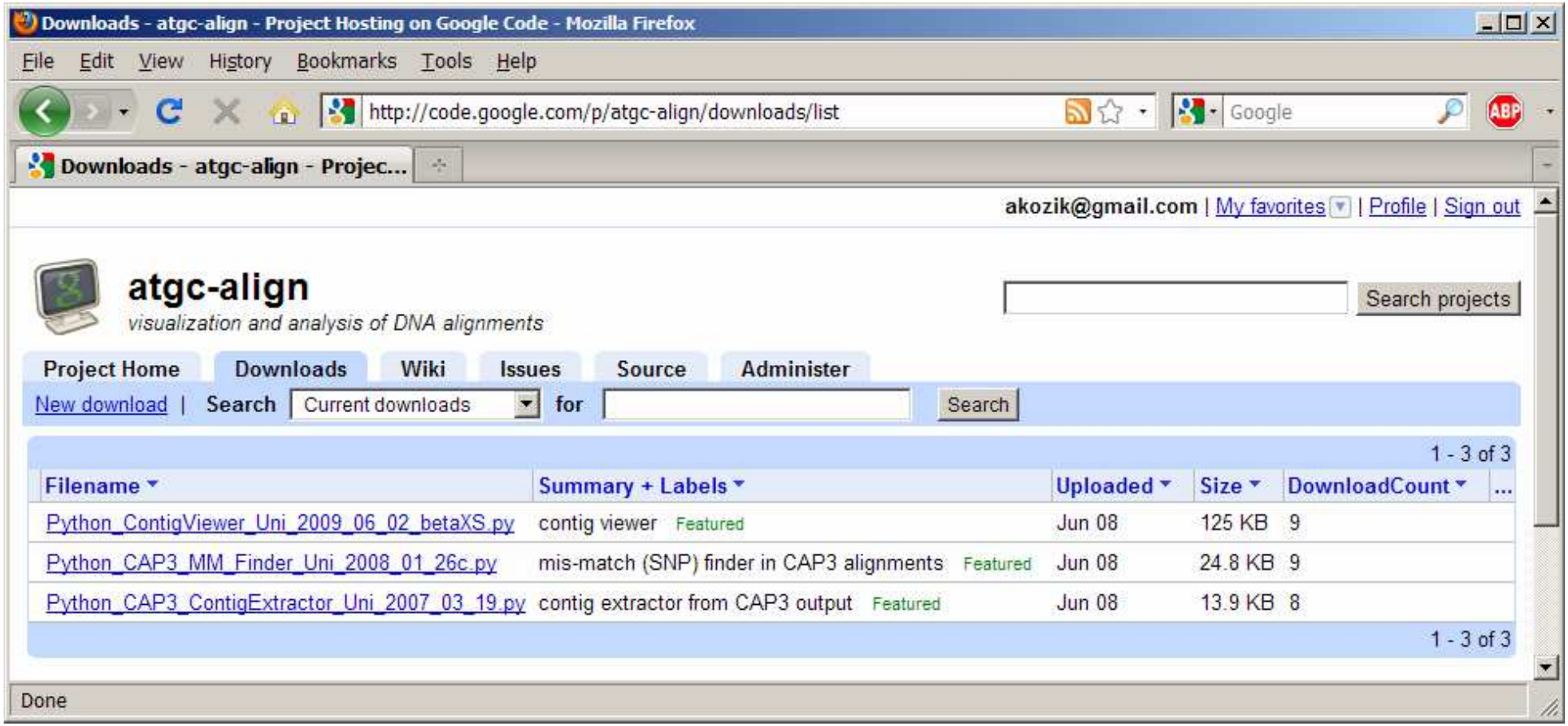
filtering for high quality reads

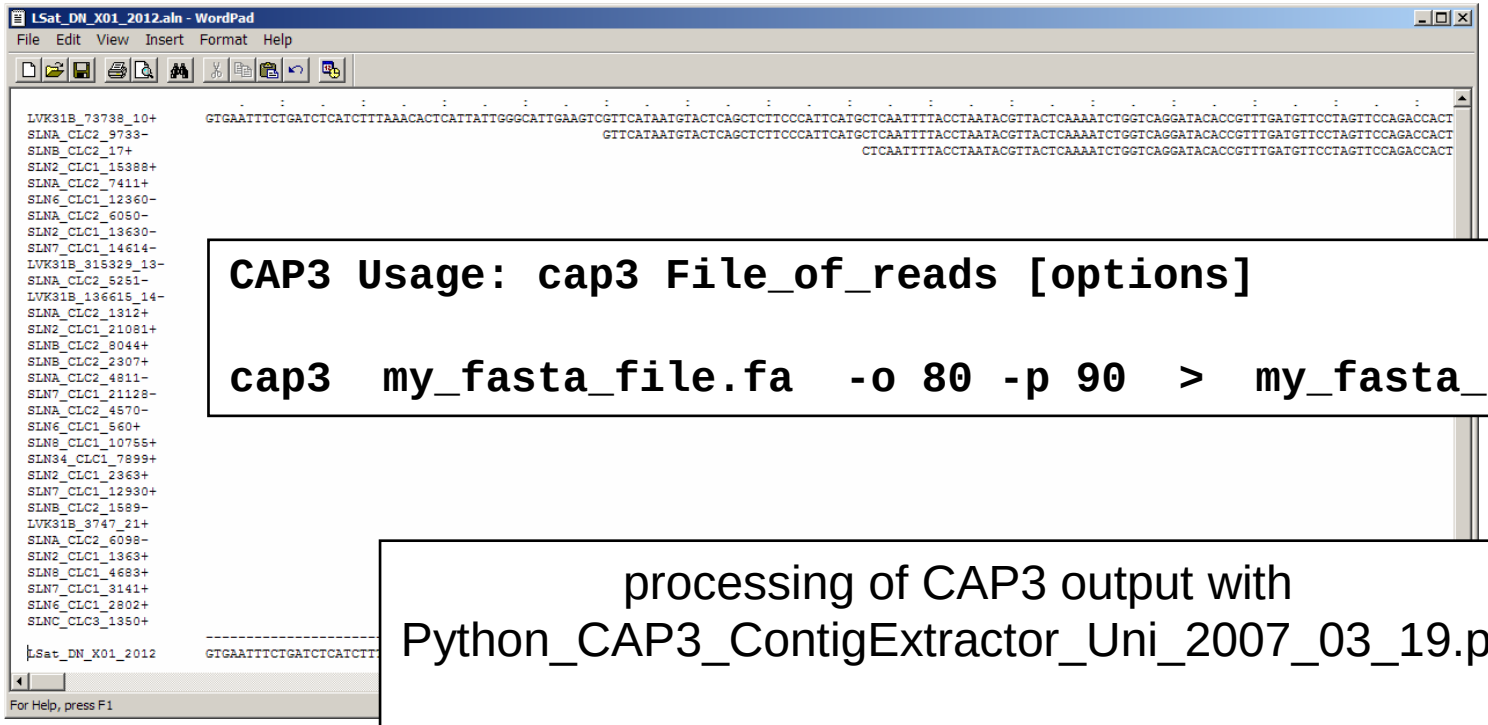
assembly parameters

visualization with Python Contig Viewer

http://code.google.com/p/atgc-align/

post-processing of CAP3 output and assembly visualization with Python Contig Viewer



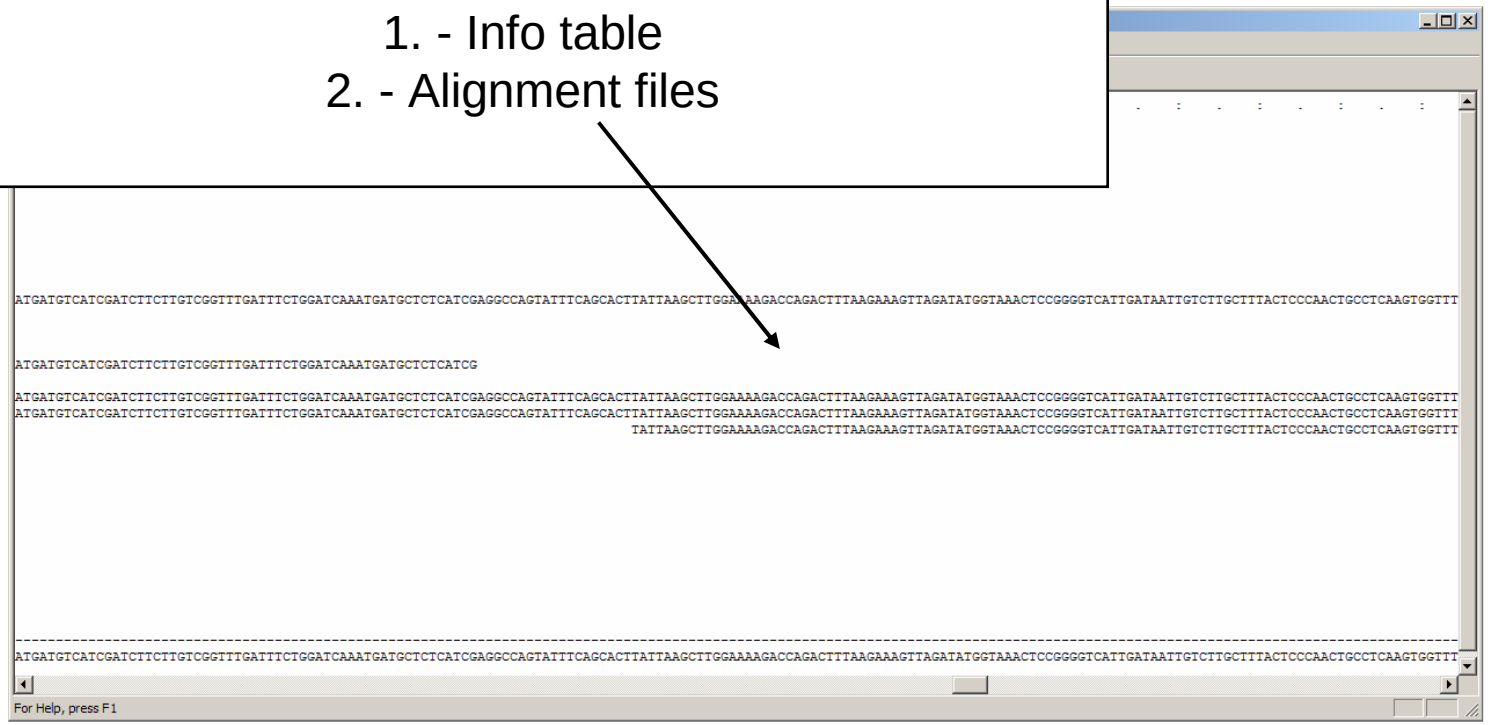


CAP3 Usage: cap3 File_of_reads [options]

cap3 my_fasta_file.fa -o 80 -p 90 > my_fasta_file.cap3.out &

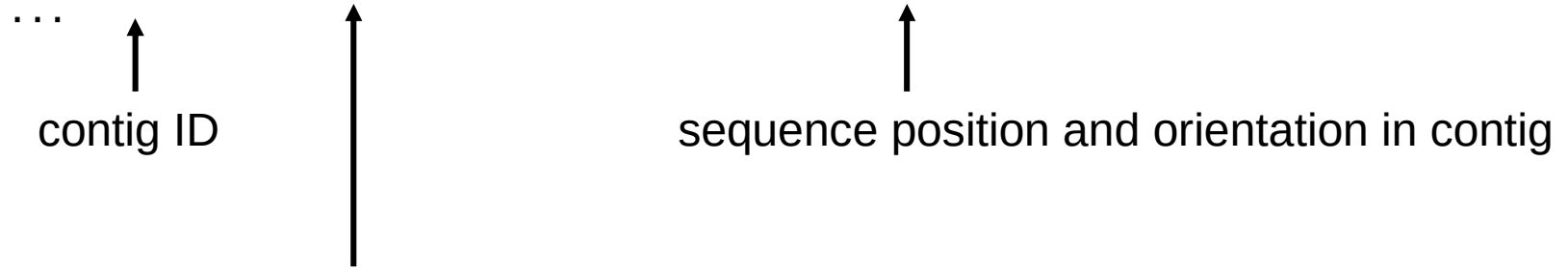
processing of CAP3 output with Python_CAP3_ContigExtractor_Uni_2007_03_19.py:

1. - Info table
2. - Alignment files



Info table - Contig Complexity information

```
SBOX_V07_10677  99      SBOD_Q65_46506_491_1-(1,521)|SBOA_Q70_63067_184_1+(11,224)|...
SBOX_V07_15959  99      SBOD_Q60_38311_405_1+(1,435)|SBOB_Q65_71217_151_6+(11,191)|...
SBOX_V07_33750  98      SBO80K31_405877_272_+(1,302)|SBO80K29_460109_274_+(1,302)|...
SBOX_V07_2653   96      SBOC_Q60_29343_843_2-(1,873)|SBOA_Q60_29209_863_2-(1,893)|...
SBOX_V07_27418  96      SBOB_Q60_23920_307_1-(1,337)|SBOC_Q60_24036_283_2-(22,332)|...
...
SBOX_V07_9935   18      SBO80K29_44764_509_4+(1,537)|SBO80K27_49894_511_4+(1,537)|...
SBOX_V07_9961   18      SBOD_Q65_112060_507_+(1,537)|SBOD_Q60_84313_508_8+(1,537)|...
SBOX_V07_10000  17      SBOC_Q60_49726_506_8+(1,536)|SBOA_Q60_49428_287_1+(11,327)|...
SBOX_V07_10018  17      SBOD_Q70_259498_506_+(1,536)|SBOB_Q65_196026_486_+(16,523)|...
...
```



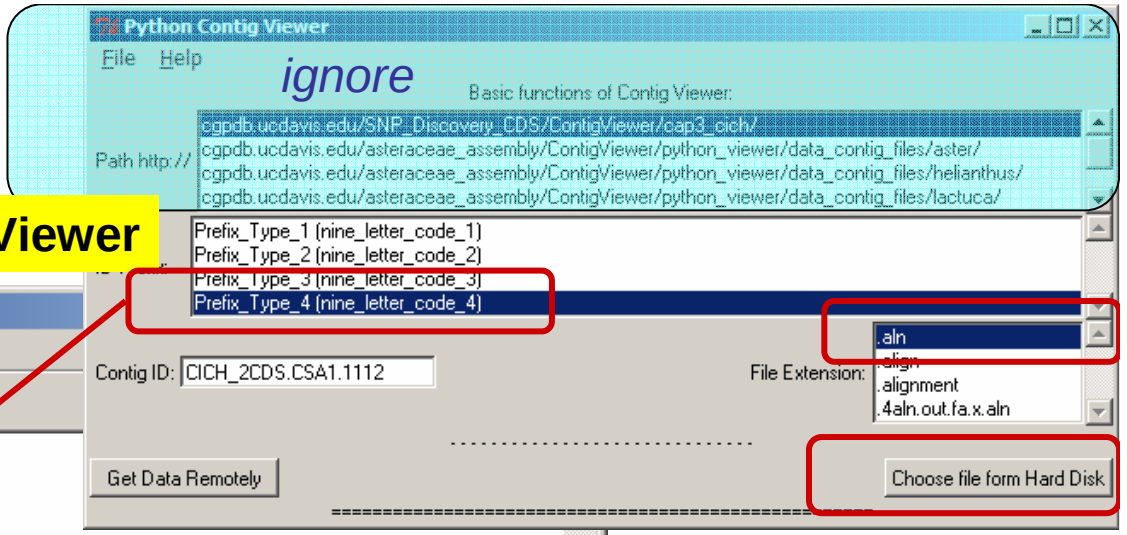
Python_CAP3_ContigExtractor_Uni_2007_03_19.py usage:

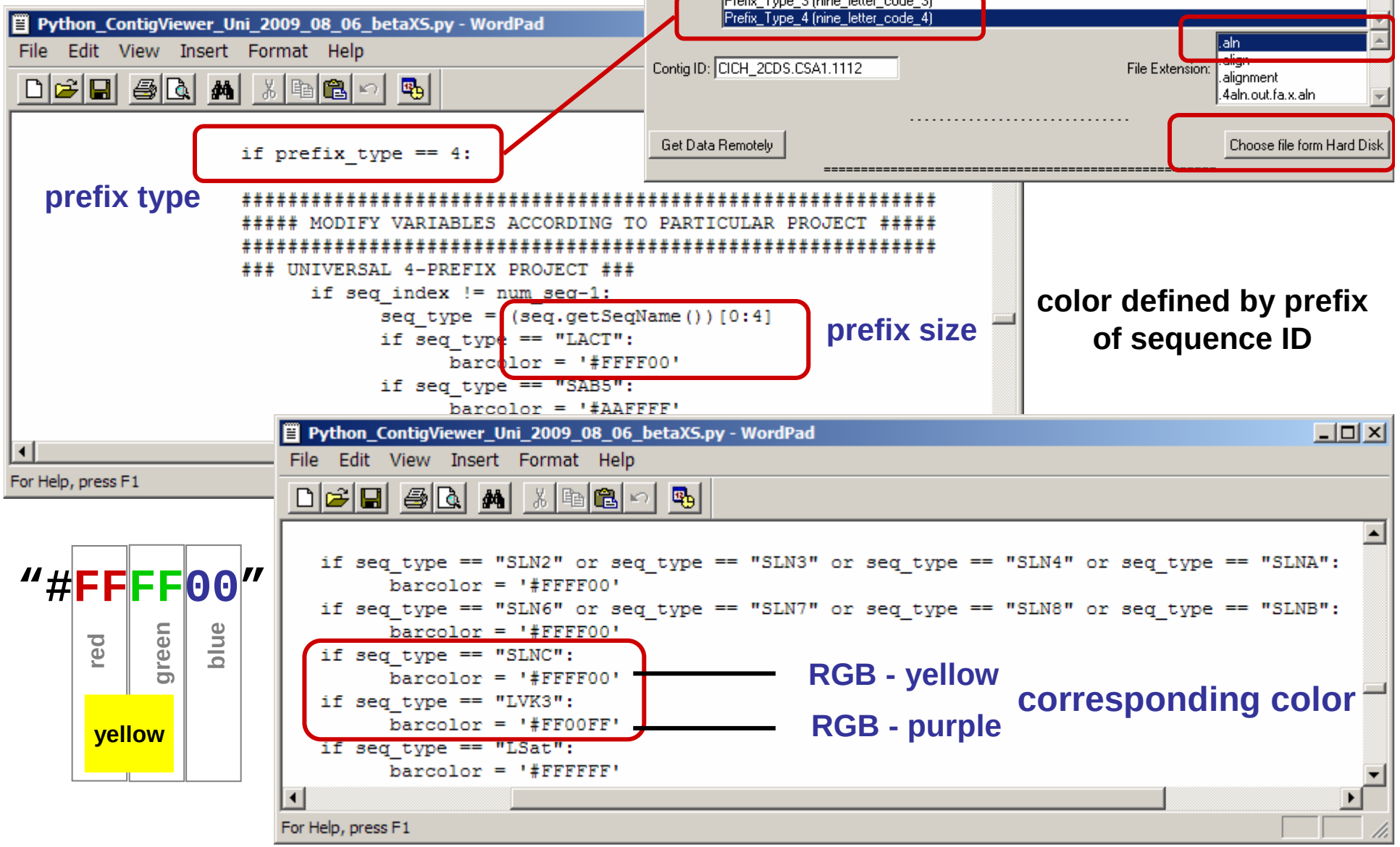
```
bash-2.03$
bash-2.03$ python Python_CAP3_ContigExtractor_Uni_2007_03_19.py
What type of output do you want? (1/2): 1
Enter the SOURCE file name: my_fasta_file.cap3.out
Enter the DESTINATION file name: my_fasta_file.cap3.out.Info
Default contig file name prefix is Contig
Enter the contig file name prefix:ASSY_V1_
read data from the input file my_fasta_file.cap3.out ...

bash-2.03$
bash-2.03$ python Python_CAP3_ContigExtractor_Uni_2007_03_19.py
What type of output do you want? (1/2): 2
Enter the SOURCE file name: my_fasta_file.cap3.out
Enter the DESTINATION directory with alignments: my_fasta_file.cap3.out.Align
Default contig file extension is aln
Enter the contig file extension :
Default contig file name prefix is Contig
Enter the contig file name prefix:ASSY_V1_
read data from the input file my_fasta_file.cap3.out ...
```

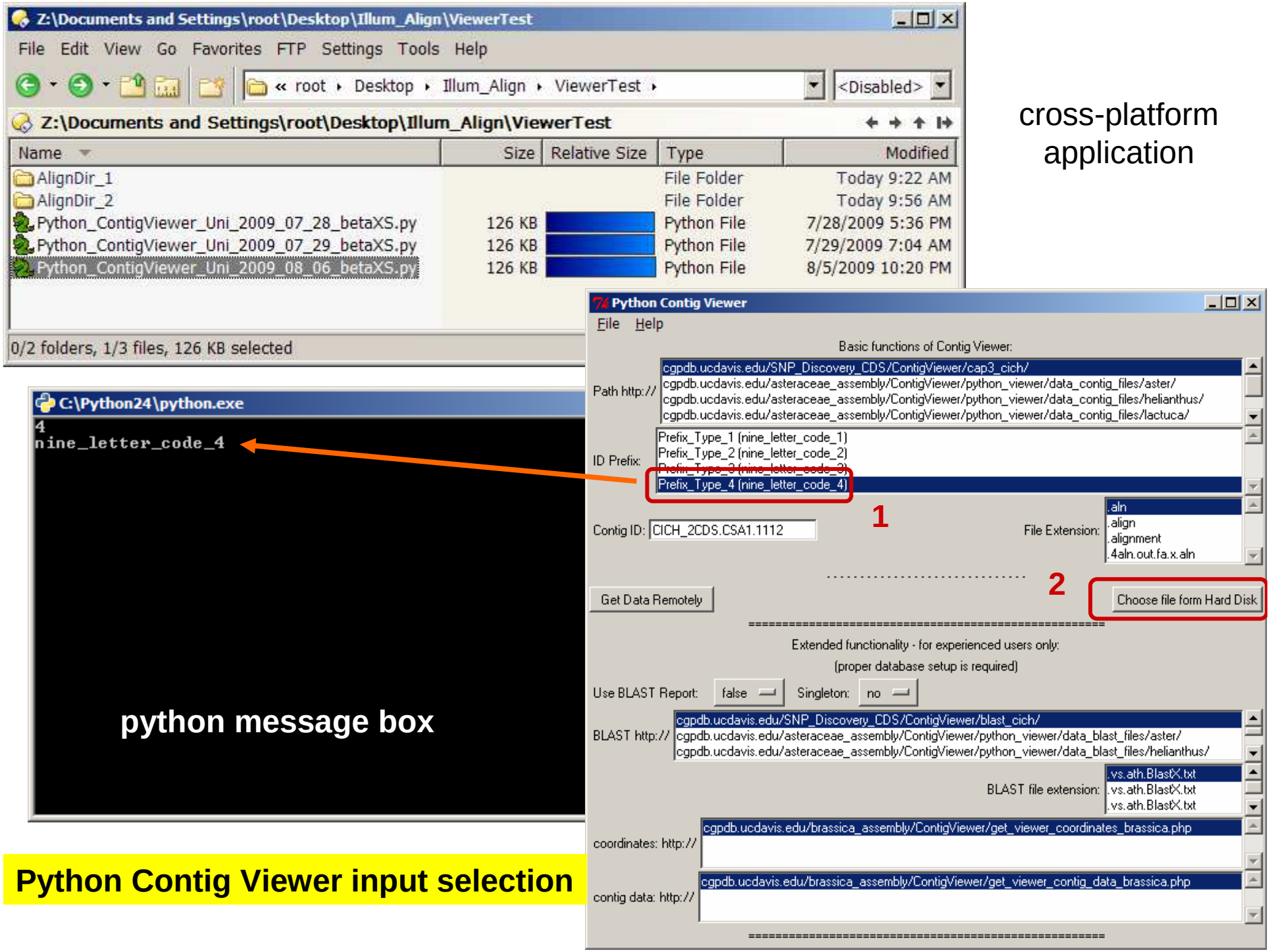
option ‘1’ to extract Info table

option ‘2’ to extract alignments

Color selection in Python Contig Viewer

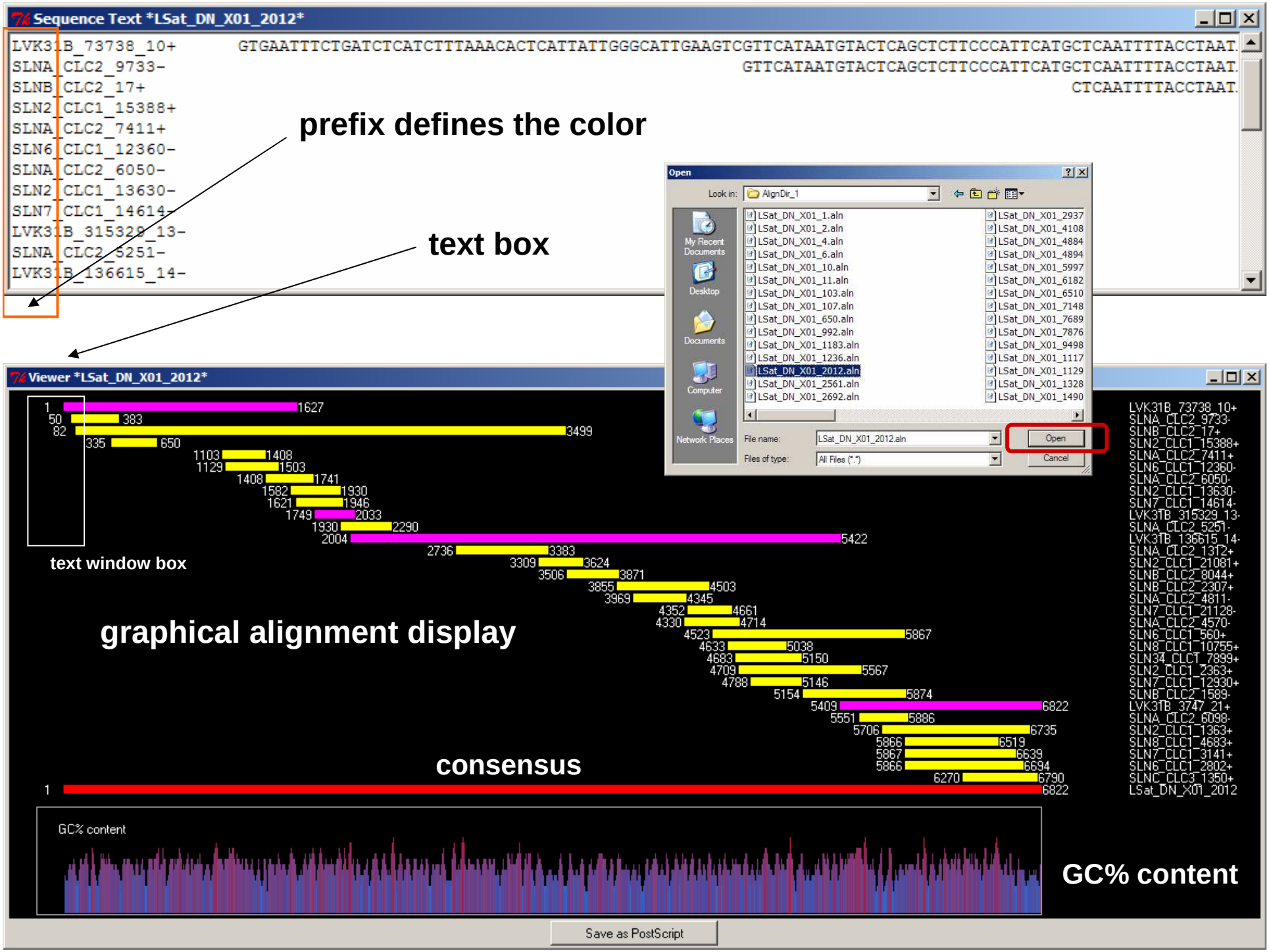






cross-platform application

Python Contig Viewer input selection


Sequence Text *LSat_DN_X01_2012*
LVK31B_73738_10+ GTGAATTTCTGATCTCATCTTTAAACACTCATTATTGGGCATTGAAGTCGTTCATAATGTACTCAGCTCTTCCCATTCATGCTCAATTTTACCTAAT.
SLNA_CLC2_9733- GTTCATAATGTACTCAGCTCTTCCCATTCATGCTCAATTTTACCTAAT.
SLNB_CLC2_17+ CTCAATTTTACCTAAT.
SLN2_CLC1_15388+
SLNA_CLC2_7411+
SLN6_CLC1_12360-
SLNA_CLC2_6050-
SLN2_CLC1_13630-
SLN7_CLC1_14614-
LVK31B_315329_13-
SLNA_CLC2_5251-
LVK31B_136615_14-
prefix defines the color
text box
Open
Look in: AlignDir_1
LSat_DN_X01_1.aln
LSat_DN_X01_2.aln
LSat_DN_X01_4.aln
LSat_DN_X01_6.aln
LSat_DN_X01_10.aln
LSat_DN_X01_11.aln
LSat_DN_X01_103.aln
LSat_DN_X01_107.aln
LSat_DN_X01_650.aln
LSat_DN_X01_992.aln
LSat_DN_X01_1183.aln
LSat_DN_X01_1236.aln
LSat_DN_X01_2012.aln
LSat_DN_X01_2561.aln
LSat_DN_X01_2692.aln
LSat_DN_X01_2937
LSat_DN_X01_4108
LSat_DN_X01_4884
LSat_DN_X01_4894
LSat_DN_X01_5997
LSat_DN_X01_6182
LSat_DN_X01_6510
LSat_DN_X01_7148
LSat_DN_X01_7689
LSat_DN_X01_7876
LSat_DN_X01_9498
LSat_DN_X01_1117
LSat_DN_X01_1129
LSat_DN_X01_1328
LSat_DN_X01_1490
My Recent Documents
Desktop
Documents
Computer
Network Places
File name: LSat_DN_X01_2012.aln
Files of type: All Files (*.*)
Open
Cancel
Viewer *LSat_DN_X01_2012*
text window box
graphical alignment display
consensus
GC% content
GC% content
Save as PostScript
LVK31B_73738_10+
SLNA_CLC2_9733-
SLNB_CLC2_17+
SLN2_CLC1_15388+
SLNA_CLC2_7411+
SLN6_CLC1_12360-
SLNA_CLC2_6050-
SLN2_CLC1_13630-
SLN7_CLC1_14614-
LVK31B_315329_13-
SLNA_CLC2_5251-
LVK31B_136615_14-
SLNA_CLC2_1312+
SLN2_CLC1_21081+
SLNB_CLC2_8044+
SLNB_CLC2_2307+
SLNA_CLC2_4811-
SLN7_CLC1_21128-
SLNA_CLC2_4570-
SLN6_CLC1_560+
SLN8_CLC1_10755+
SLN34_CLC1_7899+
SLN2_CLC1_2363+
SLN7_CLC1_12930+
SLNB_CLC2_1589-
LVK31B_3747_21+
SLNA_CLC2_6098-
SLN2_CLC1_1363+
SLN8_CLC1_4683+
SLN7_CLC1_3141+
SLN6_CLC1_2802+
SLNC_CLC3_1350+
LSat_DN_X01_2012

last remarks

UNIX tricks

fasta header modifications:

regular expressions and perl /find/replace/ function

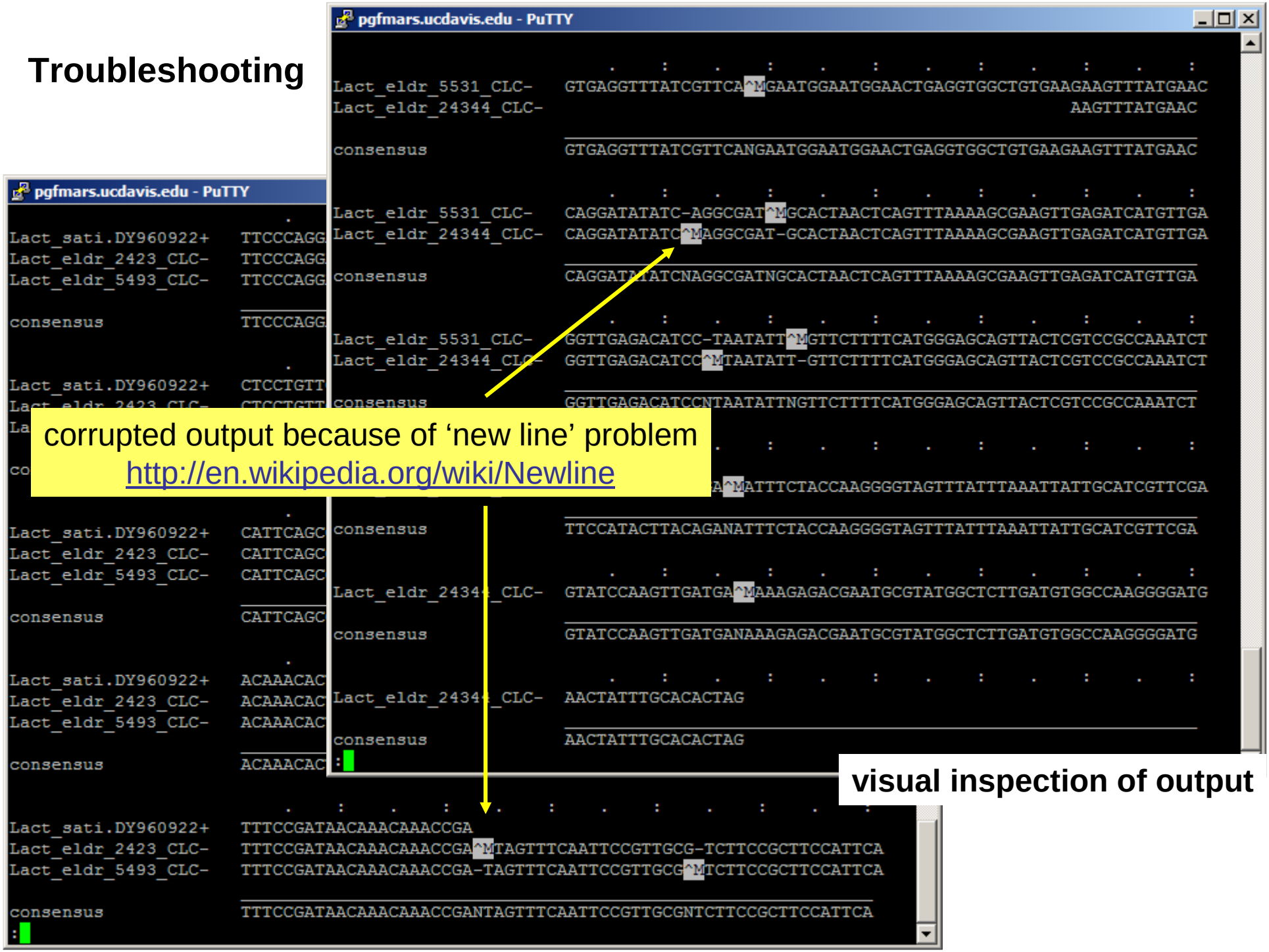
```
>NODE_24_length_783_cov_47.872288

perl -p -i -e ‘s/NODE_/SBPD_/’ my_fasta_file.txt

perl -p -i -e ‘s/_length_/_/’ my_fasta_file.txt

perl -p -i -e ‘s/_cov_/_/’ my_fasta_file.txt

perl -p -i -e ‘s/\..*//’ my_fasta_file.txt

>SBPD_24_783_47
```

trimming:

```
>SBO8_8_120_1640_1715
CCAAGAACGACATATAAACATTTTACATAATCTGGGGAACACTTGAAGAACATCAAAAGCTGGAAAAATGGATGAAAAACAGCTC

perl -p -i -e 's/\>SBO/XXXXXXXXXXXXXXX\>SBO/' my_fasta_file.orig.txt

XXXXXXXXXXXXXXX>SBO8_8_120_1640_1715
CCAAGAACGACATATAAACATTTTACATAATCTGGGGAACACTTGAAGAACATCAAAAGCTGGAAAAATGGATGAAAAACAGCTC

cut -c16-75 my_fasta_file.orig.txt > my_fasta_file.trim.txt

>SBO8_8_120_1640_1715
AAACATTTTACATAATCTGGGGAACACTTGAAGAACATCAAAAGCTGGAAAAATGGATGA
```

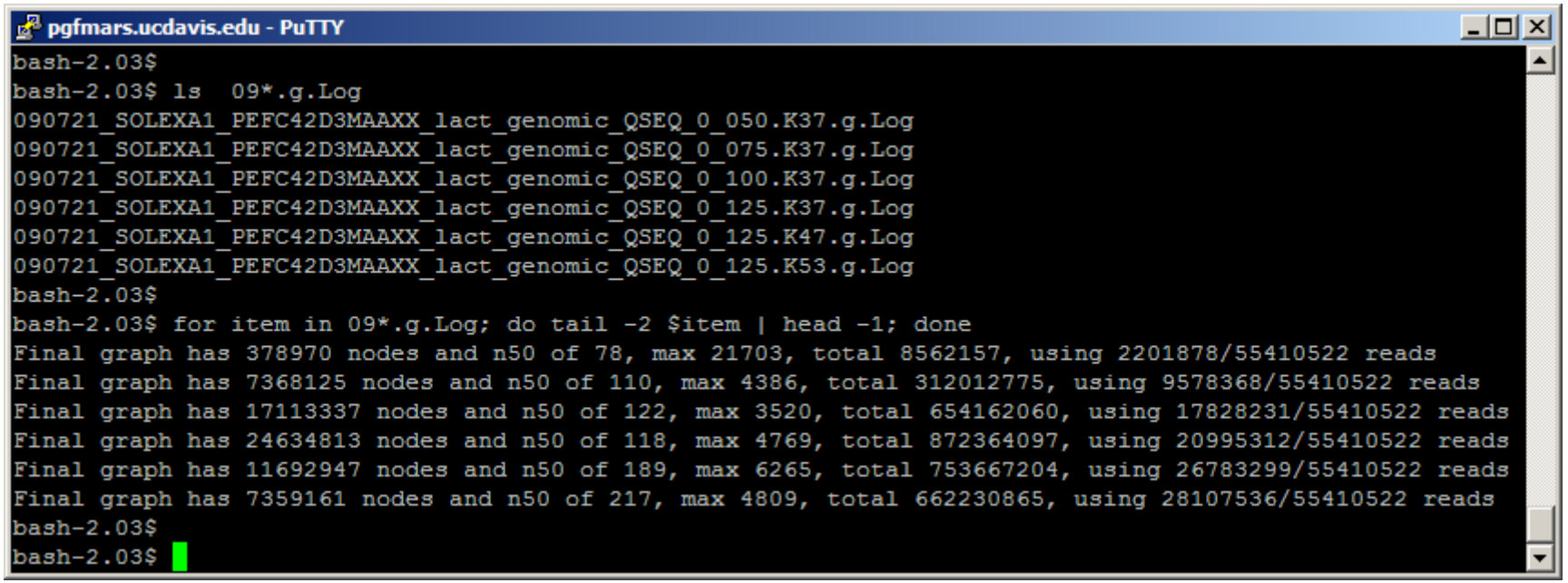
Unix ‘cut’ command

Troubleshooting



record and keep all Log files, run parameters

```
bash-2.03$ less 090721_SOLEXA1_PEFC42D3MAAXX_lact_genomic_QSEQ_0_125.K53/Log

Thu Sep  3 03:07:09 2009
 velveth.7.49.K53.exec 090721_SOLEXA1_PEFC42D3MAAXX_lact_genomic_QSEQ_0_125.K53 53 -fasta -short
090721_SOLEXA1_PEFC42D3MAAXX_lact_genomic_QSEQ_0_125.K37/Sequences

Thu Sep  3 09:02:46 2009
 velvetg.7.49.K53.exec 090721_SOLEXA1_PEFC42D3MAAXX_lact_genomic_QSEQ_0_125.K53 -min_contig_lgth 180
-unused_reads yes -amos_file yes -read_trkg yes
```




```
bash-2.03$ for item in 09*.g.Log; do tail -2 $item | head -1; done
Final graph has 378970 nodes and n50 of 78, max 21703, total 8562157, using 2201878/55410522 reads
Final graph has 7368125 nodes and n50 of 110, max 4386, total 312012775, using 9578368/55410522 reads
Final graph has 17113337 nodes and n50 of 122, max 3520, total 654162060, using 17828231/55410522 reads
Final graph has 24634813 nodes and n50 of 118, max 4769, total 872364097, using 20995312/55410522 reads
Final graph has 11692947 nodes and n50 of 189, max 6265, total 753667204, using 26783299/55410522 reads
Final graph has 7359161 nodes and n50 of 217, max 4809, total 662230865, using 28107536/55410522 reads
bash-2.03$
```

SUMMARY

ILLUMINA output data format

criteria for filtering

trimming and sampling effect on assemblies

assembly parameters

assembly validation and
exon-intron prediction by alignment

visualization with Python Contig Viewer

Acknowledgements

Marta Matvienko

UCD Genome Center Bioinformatics Core for IT support

UCD DNA Technologies Core

special credits to Huaqin Xu - database and web programmer
Brian Chan - initial work with Python Contig Viewer

Richard Michelmore

SEQanswers forum and Velvet newsgroup