Workshop session MIRA

Getting a feel for MIRA and sequencing data: tips & tricks

Bastien Chevreux – DSM Nutritional Products

Workshop Comparative Genomics

Český Krumlov, Jan. 2011

Testing that MIRA and other programs run

No need to do this on the Linux images, it simply works there (thanks, Adam!)

- Test mira, caf2gap, gap2caf
  - At the command prompt, type:
    `$ which mira`
    - You get something like: `/usr/local/bin/mira`
    - If there is no output, you have a problem!
  - Do the same with **`which caf2gap`** and **`which gap2caf`**
- gap4
  - check whether STADENROOT is set:
    `$ echo $STADENROOT`
  - **ONLY** if no result, set STADENROOT
    `$ export STADENROOT="/path/the/staff/gives/you"`
  - Source in the basic gap4 environment:
    `$ . $STADENROOT/staden.profile` (yes, that is a dot at the start of the line)
  - you should be able to start gap4 and and get a window
    `$ gap4`

General Sanger pre-processing workflow

Data acquisition	`.ab1`
↓	↓
Extraction of basic sequence and quality data	`.FASTA, .FASTA quality, .phd, .exp, .scf`
↓	↓
Adding ancillary data	`.exp, .scf, .phd, XML Traceinfo`

Basic data vs. Ancillary data

- Basic data
  - sequence
- Ancillary data
  - sequence quality

Basic data vs. Ancillary data

- Basic data
  - sequence
  - sequence quality
- Ancillary data

Never, ever assemble without quality values

I don't care whether

- your multi-million dollar sequencing project produces junk
- and your institute is discredited
- and you loose your job
- and your GF/BF runs away
- and you become alcoholic
- and homeless
- and die without friends ...

... just because you assembled **without** quality values ...

... but - please - be nice and think of the kitties!

Basic data vs. Ancillary data

- Basic data
  - sequence
  - sequence quality
- Ancillary data
  - quality clipping
  - clipping of unwanted data (sequencing vectors, adaptors, etc.)
  - info on sequencing method
  - info on chemistry used
  - library information (insert size, etc.)
  - ...
  - ...
  - ...

Clipping / trimming, masking


```
>someread
ACGTAGACGCTCAGTACTGTTTGGTGTCAGTGTGTACGTGT
```

Assume these two being “unwanted”
(bad quality or whatever)

How to convey that information to the assembler?

Hard clipping / trimming


```
>someread
ACGTAGACGCTCAGTACTGTTTGGTGTCAGTGTGTACGTGT
```

```
>someread
ACGCTCAGTACTGTTTGGTGT
```

Positive:

- Save space
- No confusion possible

Negative:

- Lose clipped sequence
- Lose linkage to n-th base in original sequence

Masking


```
>someread
ACGTAGACGCTCAGTACTGTTTGGTGTCAGTGTGTACGTGT
```

```
>someread
xxxxxx                                  xxxxxxxxxxxxxxx
```

Positive:

- No confusion possible
- Keeps linkage to n-th base in orginal sequence

Negative:

- Needs more space
- Lose clipped sequence

Soft clipping (2)
Using ancillary information


```
>someread
ACGTAGACGCTCAGTACTGTTTGGTGTCAGTGTGTACGTGT
```

```
>someread
ACGTAGACGCTCAGTACTGTTTGGTGTCAGTGTGTACGTGT
```

Clip left: pos. 7

Clip right: pos. 28

Positive:

- Adds info, discards nothing

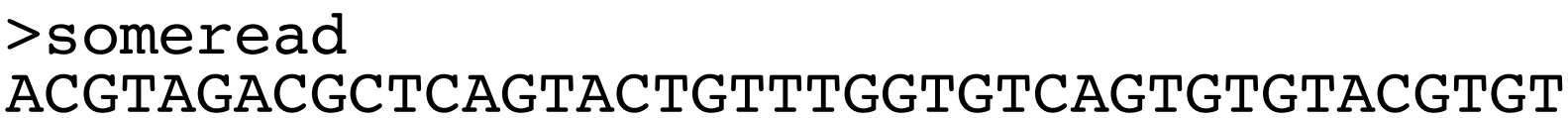
Negative:

- Needs even more space / additional files
- Not all programs can read all ancillary file formats
- Convention needed on coordinate system 0 or 1 based, including / excluding, etc.)

Soft clipping (3)
Clipping for all sorts of things

Qual left: pos. 7

Qual right: pos. 28
Vector / adaptor right: pos. 23



Small EXP example

Note: not all fields are mandatory and many more fields possible


```
ID   U13e07f12.t2
LN   U13e07f12.t2.scf
LT   SCF
QR   944
SQ
     GGGCTGCGAG GCTGTTCGTG CATCCTGCAG GTCGACTCTA GAGGACCCCA GTATGAGCAA
...
     ATTAATATGT TTACCCCTCT CACCTTTTTG GGCCAATTAG GTT
//
DT    4-Sep-2002
OP   bach
QL   47
QR   613
SL   69
SF   /gen/pck/gist/lib/vectordata/m13mp18.fas
```

Read identifier

File with raw data

Raw data file type

Qual clip right

Sequence

Processing date

Operated by whom

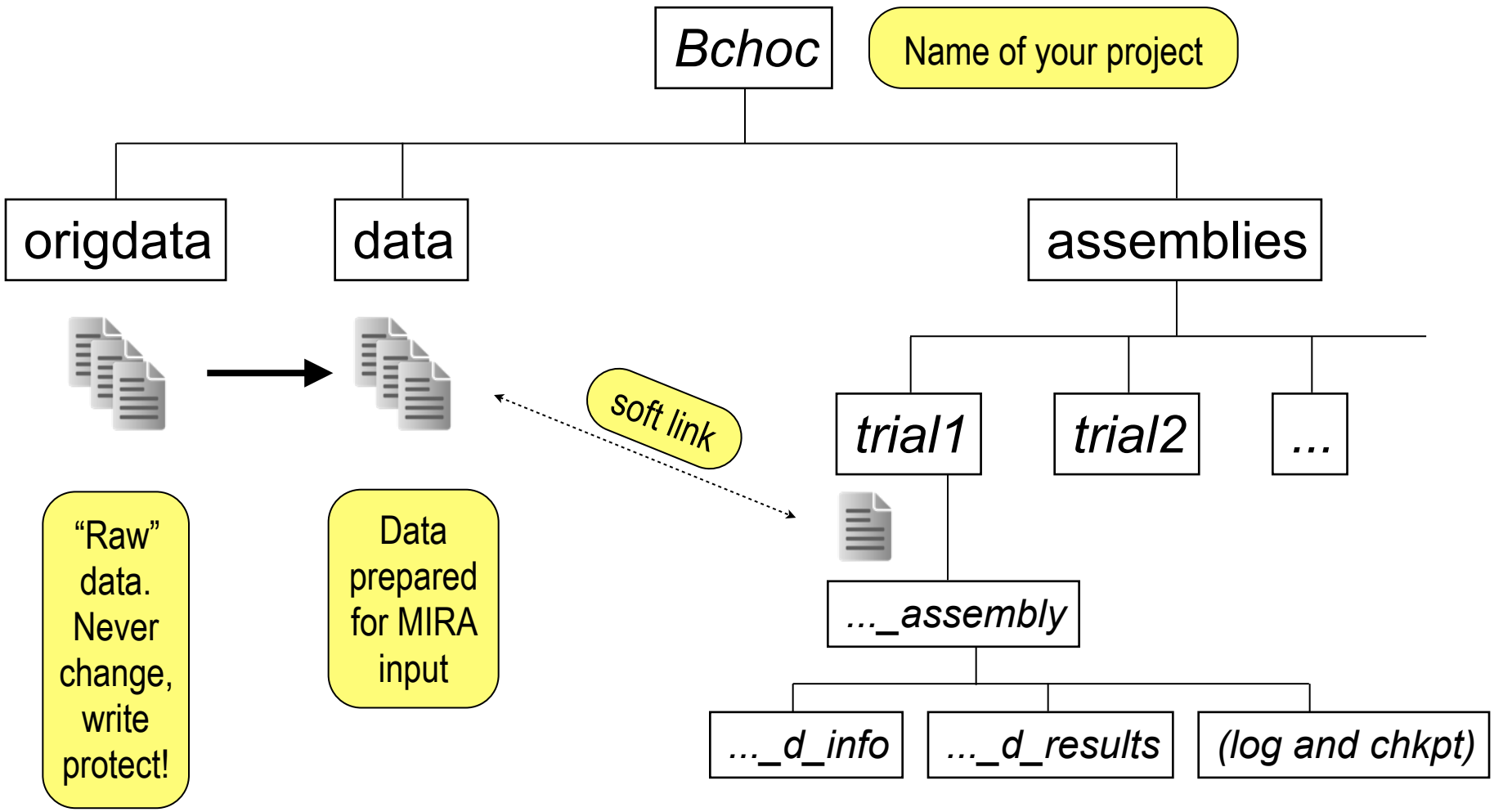
Qual clip left
Qual clip right (another)

Sequencing vector clip left

File with sequencing vector found

Proposal for project structure

Organise your files the way you want, but organise! And (try to) be consistent.



Exercise/demo data sets for you

- 7 data sets, > 8 GiB of (packed!) data (~ 37 GiB unpacked)
- one archive per data set
  - unpack via: `$ tar xvjf name_of_dataset.tar.bz2`
  - for assemblies: structure set up mostly as in previous slide
    - origdata: if present, the original data plus pointers where downloaded
    - data: data after pre-processing, eventually just the scripts which do that
    - assemblies: every subdirectory, has a ready-to-run **`runme.sh`** script for MIRA
- on sticks: small data sets only!
  - 3 exercises: 01, 02 and 07: Sanger de-novo, mapping. Eukaryote coverage.
  - suited for everyone and can run even on the smallest netbook
- ask staff how to get full data sets!
  - 4 demo data sets: 03, 04, 05, 06: Solexa mapping, 454 & Solexa de-novo and mapping, data problems

Exercise: 01_U13_sanger_denovo (1)
A “simple” 36kb BAC project

- unpack and go to the project directory
- take a look at the `data` directory
  - take a look at the `scf` directory
    - if you have Linux and Staden, try: `$ trev U13e07f12.t2.scf`
  - take a look at the `exp_orig` directory:
    - try: `$ less U13e07f12.t2.exp`
- go to `assemblies/firsttest`
  - look at the run script: `$ less runme.sh`
  - run MIRA: `$ bash runme.sh`
  - go to the newly created directory with MIRA output: `$ cd U13_assembly`
  - look around: `$ ls`
  - go to info directory and lear about basic info files: `$ cd U13_d_info`

```
$ less U13_info_assembly.txt
$ less U13_info_contigstats.txt
$ less U13_info_repeats.txt
$ ...
```

Exercise: 01_U13_sanger_denovo (2)
A “simple” 36kb BAC project

- go to the `U13_d_results` directory
- convert the CAF file to a Staden gap4 database:
  `$ caf2gap -project U13 -ace U13_out.caf`
- if you have not already done so, initialise Staden environment:
  `$ . $STADENROOT/staden.profile`
- start gap4 with the database
  `$ gap4 U13.0`
- follow course instructions on screen to learn basics of gap4 and MIRA data
  - how MIRA disentangled repeats ...

Exercise: 01_U13_sanger_denovo (3)
A “simple” 36kb BAC project

- go to assemblies/secondtest
- look at the run script: $ **less runme.sh**
  - what are the differences to the script in “firsttest”?
  - what are the differences of the EXP files linked?
- run MIRA: $ **bash runme.sh**
- in the info directory, look at the basic assembly information
  $ **less U13_info_assembly.txt**
  $ **less U13_info_contigstats.txt**
- in the results directory, create a gap4 database and look at it
  $ **caf2gap -project U13 -ace U13_out.caf**
  $ **gap4 U13.0**
- Follow on-screen course instructions to see hidden problems of that project

Exercise: 02_cje_sanger_backbone (1)
Mapping Sanger to the first 40kb of *Campylobacter jejuni*

- unpack and go to the project directory
- a look at the data directory
  - GenBank file
    - first 40kb of C. jejuni
    - fully annotated
  - Sequence & quality data
    - FASTA and FASTA quality format
  - Strain data
    - which read comes from which strain
  - Ancillary data
    - Traceinfo format
    - XML
    - Standardised by the NCBI

```
LOCUS       NC_002163        40020 bp                    EP-2005
ACCESSION   NC_002163
VERSION     NC_002163.1  GI:
FEATURES             Locat
     source          1..40020
                     /organism="Campylobacter j
                     /strain="NCTC 11168"
     gene            1..1323
                     /gene="dnaA"
                     /locus_tag="Cj0001"
                     /db_xref="GeneID:904342
     CDS             1..1323
                     /gene="dnaA"
                     /locus_tag="Cj0001"
                     /note="dnaA, probabl
                     initiator protein"
     gene            complement(15844..16419)
                     /gene="rnhB"
                     /locus_tag="Cj0010c"
                     /db_xref="GeneID:904334"
     CDS             complement(15844..16419)
```

Header: basic info

Feature section: “sticky notes” on the sequence

Source: what is this, how long, etc.?

Gene: from and to coordinates; gene name; formal name (locus tag); cross-references etc.

CoDing Sequence (protein): from and to coordinates; protein name; formal name (locus tag); annotation notes etc.

Gene: another gene, this time in reverse direction

MIRA strain data file
A simple key value file

(actually not needed when having just one reference strain and one sequencing strain)

```
NC_002163     cjejuniNCTC11168
GCJAA15TF     cjejuniRM1221
GCJAA49TF     cjejuniRM1221
GCJAA63TF     cjejuniRM1221
GCJBB08TF     cjejuniRM1221
GCJBB08TR     cjejuniRM1221
GCJBB11TF     cjejuniRM1221
...
```

Backbone (reference) read info

Reads of the sequencing project

Read name

Strain the read belongs to

```
<trace>
  <trace_name>GCJAA15TF</trace_name>
  <template_id>GCJAA15</template_id>
  <insert_stdev>600</insert_stdev>
  <insert_size>2000</insert_size>
...
  <center_name>TIGR</center_name>
  <species_code>CAMPYLOBACTER JEJUNI RM122
  <source_type>GENOMIC</source_type>
  <trace_type_code>WGS</trace_type_code>
...
  <plate_id>GCJAA</plate_id>
  <well_id>B3</well_id>
...
  <chemistry_type>TERMINATOR</chemistry_type>
  <program_id>PHRED (0.990722.G) AND TTUNER (1.
...
  <svector_code>PHOS2</svector_code>
  <clip_quality_left>3</clip_quality_left>
  <clip_quality_right>622</clip_quality_right>
  <clip_vector_left>1</clip_vector_left>
  <clip_vector_right>944</clip_vector_right>
</trace>
```

Identifiers: read and template ID
Library info: template insert size and standard deviation

Sequencing Center: TIGR
What: genomic DNA, Cjejuni RM1221
How: Whole Genome Shotgun

Info for sample tracking: plate and well ID

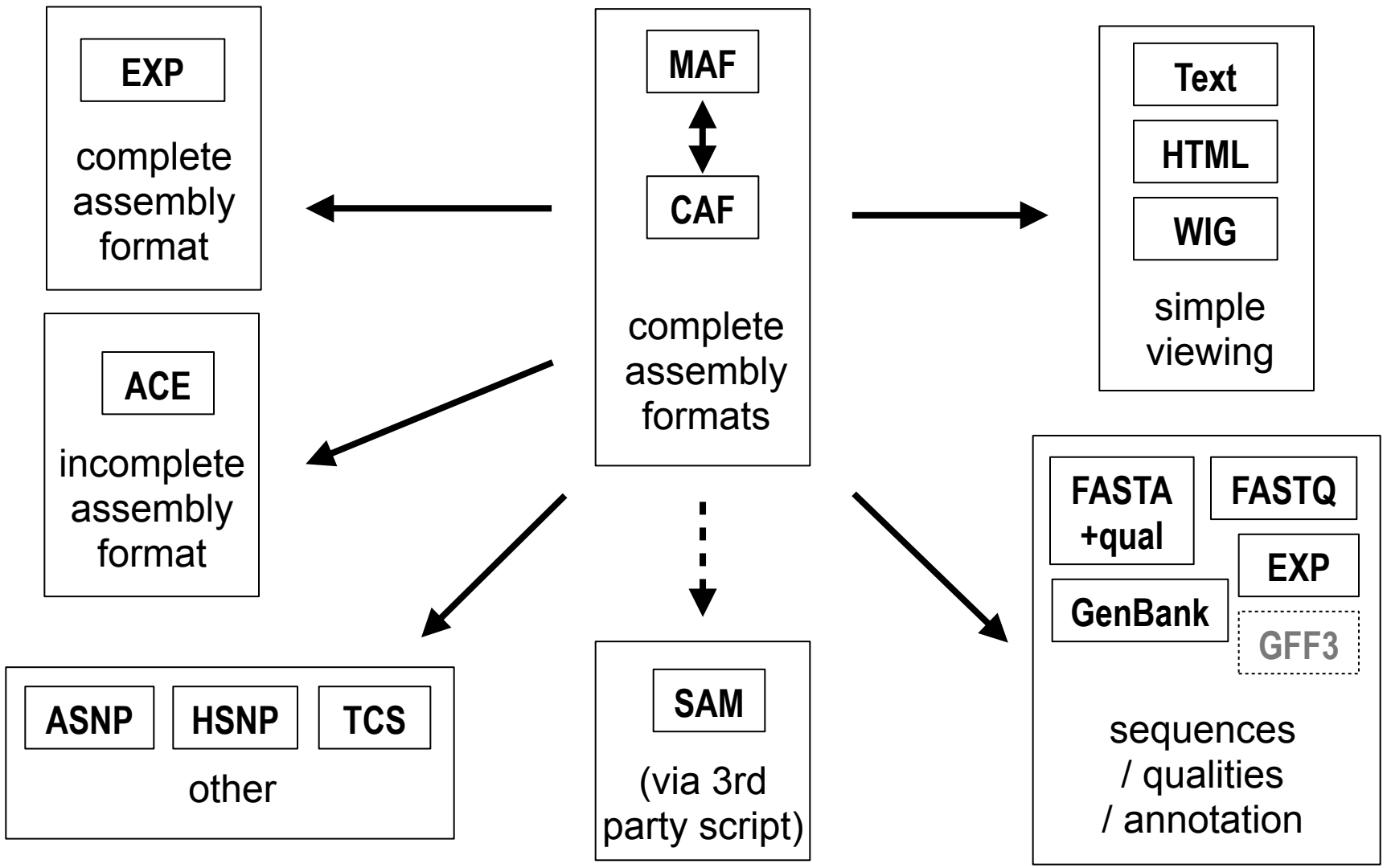
Info for base calling: chemistry and programs used

Clipping info:
sequencing vector used/found; left and right quality and sequencing vector clips

Exercise: 02_cje_sanger_backbone (2)
Mapping Sanger to the first 40kb of *Campylobacter jejuni*

- go to assemblies/firsttest
  - look at the run script: $ **less runme.sh**
  - look at the parameter file used: $ **less parameters.par**
- run MIRA: $ **bash runme.sh**
- after the run:
  - have a look at the info files in the info directory
  - in the results directory, convert the CAF to a gap4 database and start gap4
    $ **caf2gap -project cjejuni_demo -ace cjejuni_demo_out.ace**
    $ **gap4 CJEJUNI_DEMO.0**
- follow course directions to learn about “tags” in MIRA and gap4
  - the gap4 “Search” function
  - standard “GenBank” Features
  - MIRA tags in reads and consensus for strain comparison:
    - SROc/SROr, MCVc

convert_project
Conversion possibilities

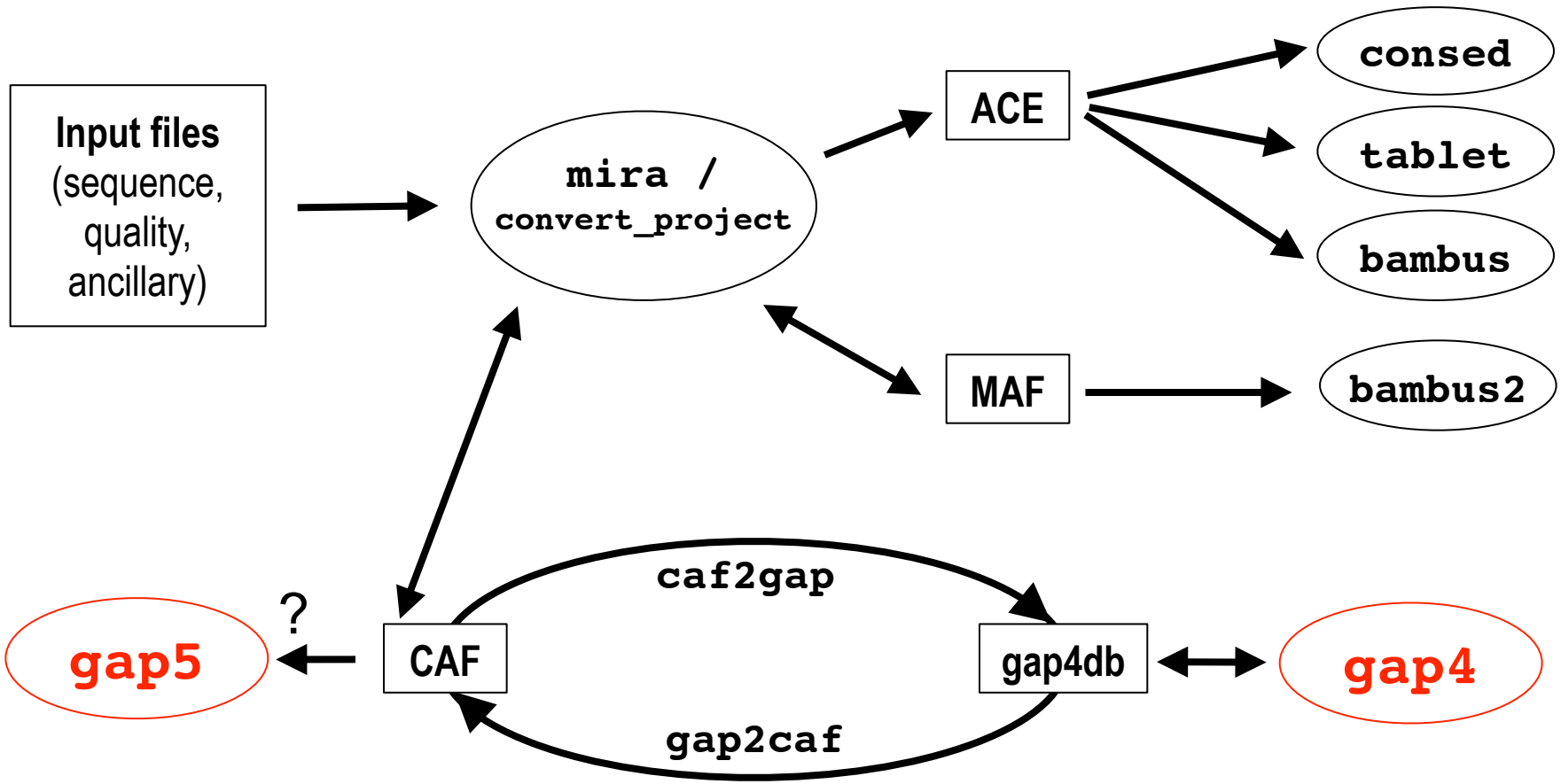


Exercise: 02_cje_sanger_backbone (3)
Mapping Sanger to the first 40kb of *Campylobacter jejuni*

- learn about the `convert_project` utility from the MIRA package
- in the results directory:
  - call `convert_project` without parameters
  - convert either the CAF or MAF to HTML output showing SNP and deletion positions, e.g.:
    `$ convert_project -f maf -t hsnp cjejuni_demo_out.caf cjejuni_demo`
  - look at the HTML file a a browser, e.g.:
    `$ konqueror cjejuni_demo_info_snpenvironment.html`
  - have MIRA perform a simple SNP analysis, e.g. from CAF:
    `$ convert_project -f maf -t hsnp cjejuni_demo_out.caf cjejuni_demo`

Possible post-processing

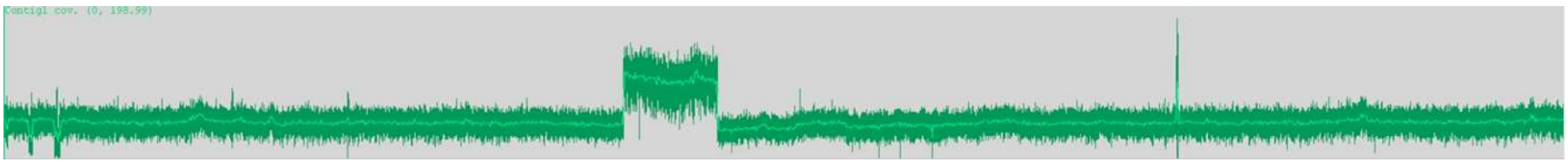
MIRA may not be the end of the line ...



IGB / GenoViz

Getting a quick view at the coverage

Remember this?



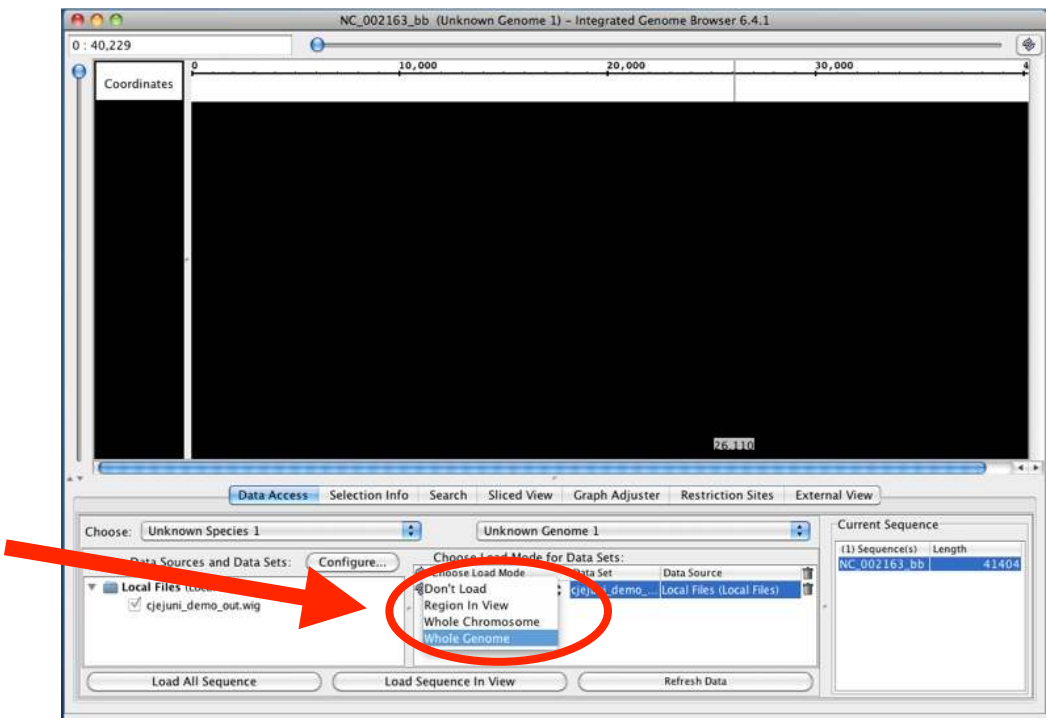
- IGB (Integrated Genome Browser) / GenoViz
- data file displayed:
  - WIG (wiggle)
  - values (integer or float), one entry per line
  - + some general description
  - ideal to store, e.g., coverage data: one value per consensus base
- to start IGB, use the “run_igb.sh” script from IGB distribution

```
$ run_igb.sh
```

Exercise: IGB / GenoViz (1)

Looking at coverage data of *Campylobacter jejuni*

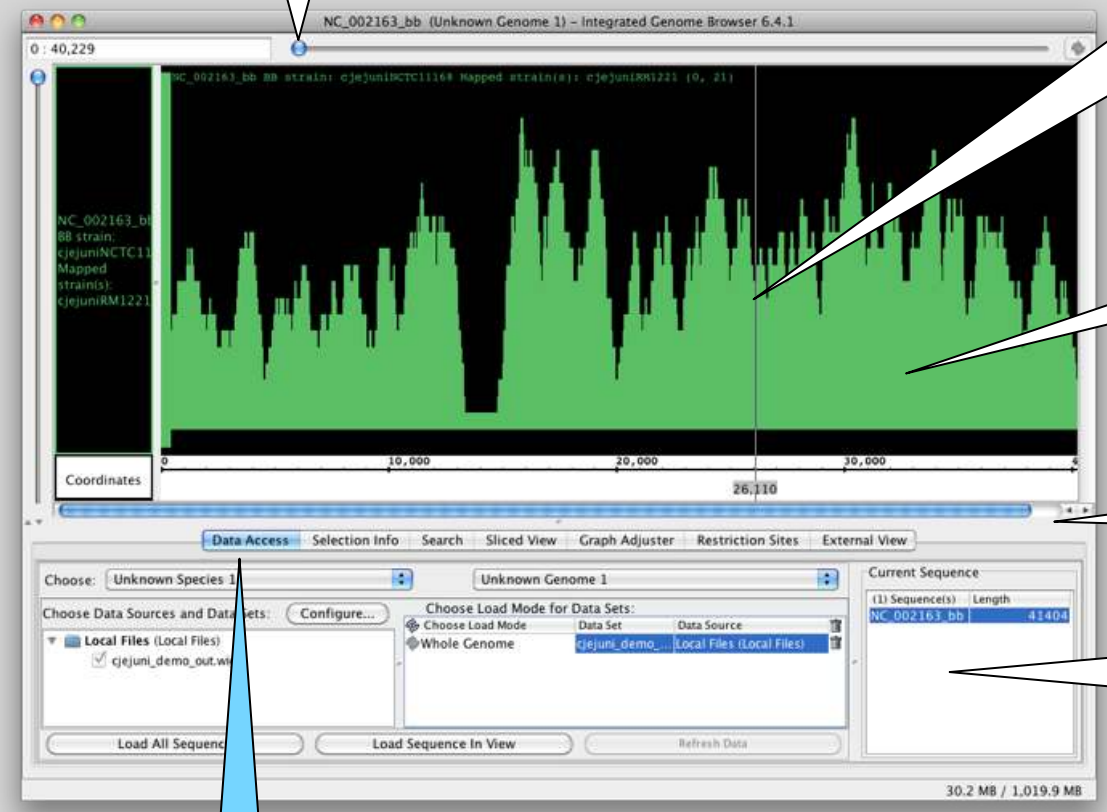
- basics on how to use IGB / GenoViz ...
- Use menu entry:
  File ➔ Open File
- Navigate to results directory of 02_cje_...
- Load the .wig wiggle file
- Mind the almost hidden pop-up appearing! (circled area)
  - Click to open pop-up, change “Region in view” to “Whole Genome”



Exercise: IGB / GenoViz (2)

Looking at coverage data of *Campylobacter jejuni*

Slider: Zoom X direction (always visible)



Cursor (vertical line)

click to set

Coverage data

Scrollbar: X axis

always visible

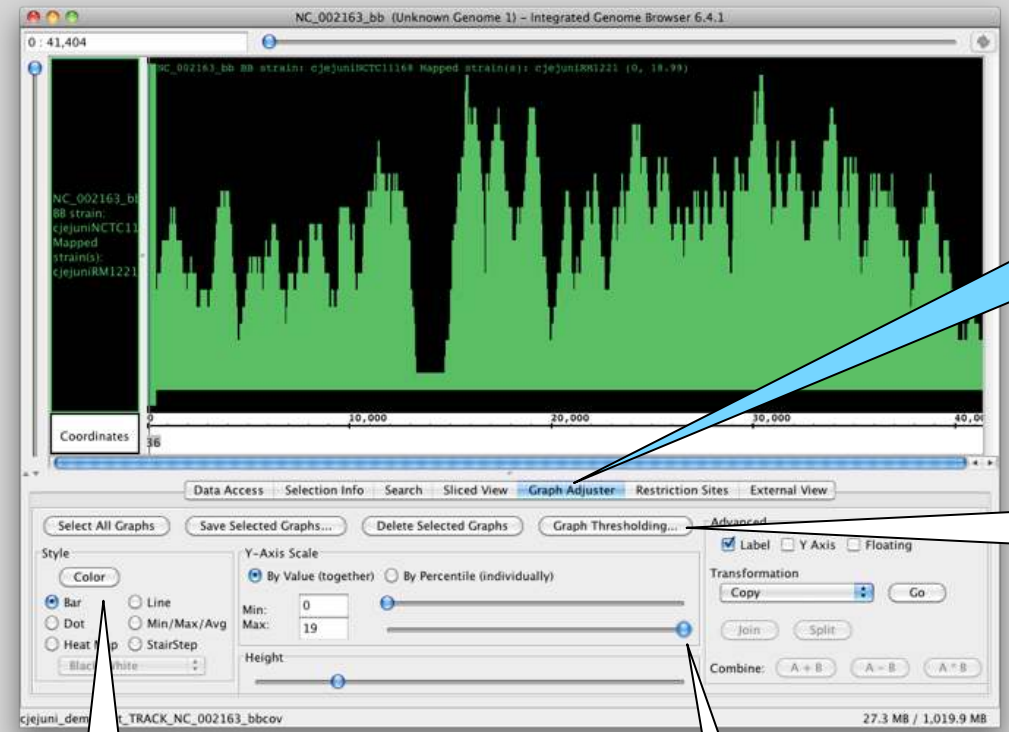
Chooser: Contig sequence displayed (only 1 in this example)

Only on “Data access” tab!

Tab chooser: Data access

Exercise: IGB / GenoViz (3)

Looking at coverage data of *Campylobacter jejuni*



Tab chooser: Graph adjuster

Button: Graph thresholding

functionality to set and display thresholds

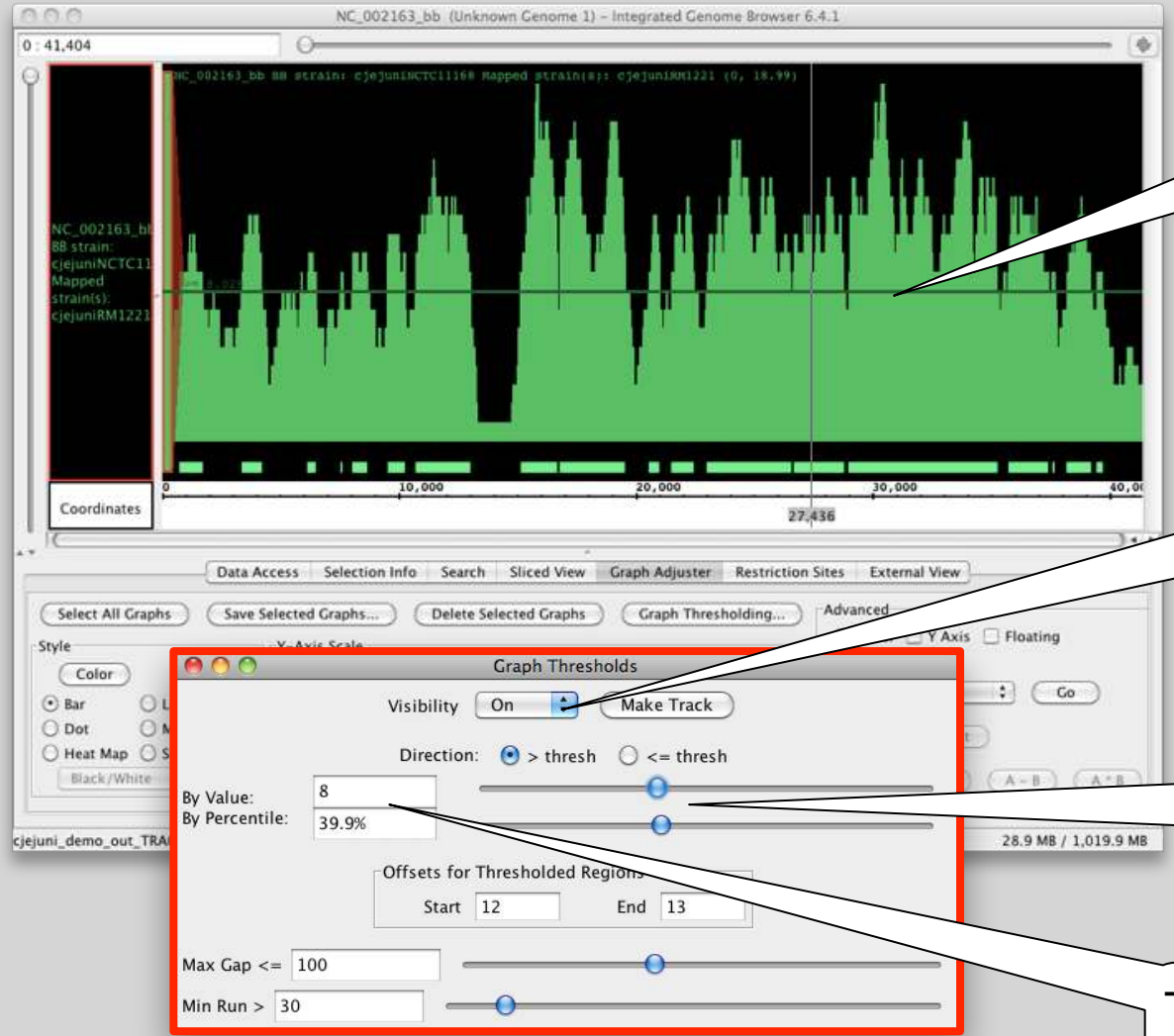
Graph display style chooser

Most useful (although not in this example): instead “Bar”, use “Min/Max/Avg”

Slider Zoom: Y-direction

Exercise: IGB / GenoViz (4)

Looking at coverage data of *Campylobacter jejuni*



Line: current threshold

Button: Threshold visibility

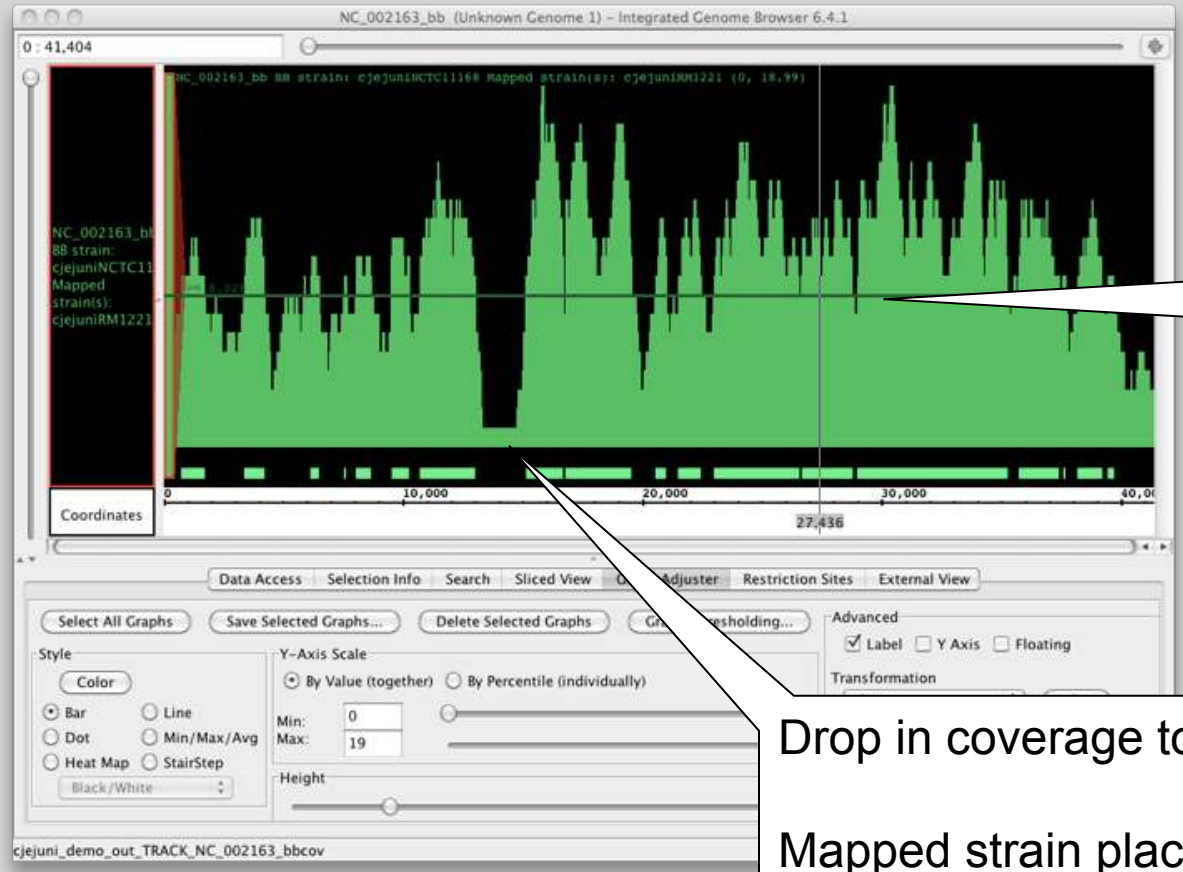
Choose “On”

Sliders: choose threshold level

Text boxes: enter threshold

Exercise: IGB / GenoViz (5)

Looking at coverage data of *Campylobacter jejuni*



Line: current threshold

set to 8

Almost all the project has “normal coverage, but ...

Drop in coverage to 0

Mapped strain placed no reads! Possible deletion or major difference.

In MIRA/gap4 look out for MCVc and SROc tags!

Exercise: 07_eukaryot_wig (1)

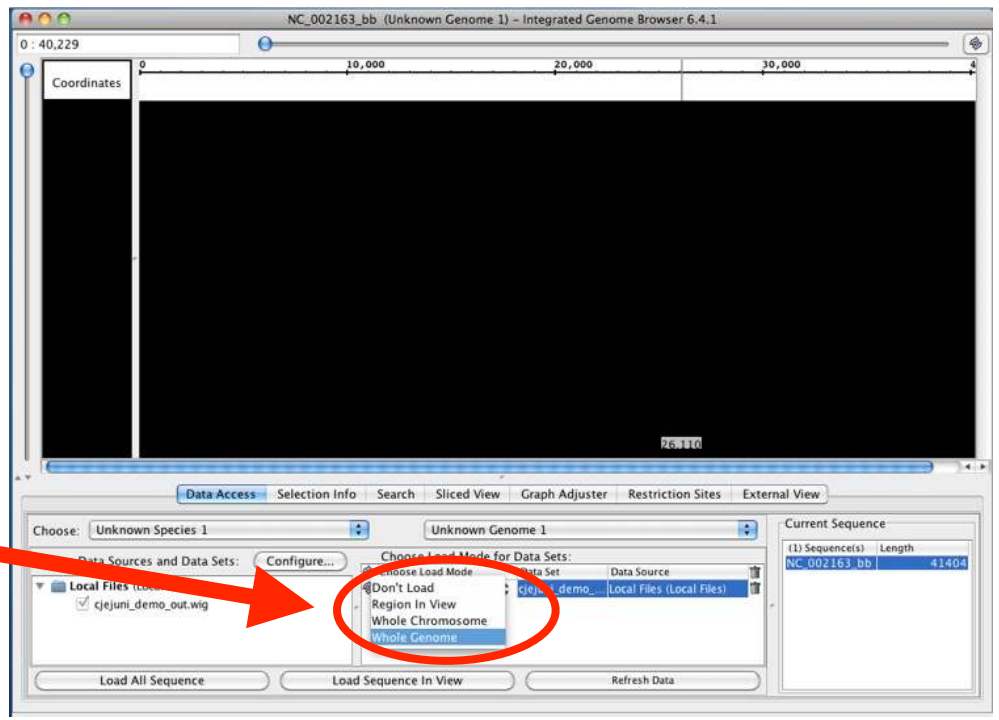
Eukaryotes are mean ... even in mapping assemblies

- A “simple” mapping experiment
- lower eukaryote (fungus)
  - DNA sequence: several chromosomes and one mitochondrion
- Sequencing done with Solexa
  - 36 bp, single reads (not paired-end, no mate-pairs)
  - enough for ~28x coverage
- Partial results file
  - only coverage data (.wig)
  - 4 chromosomes + the mitochondrion

Exercise: 07_eukaryot_wig (2)

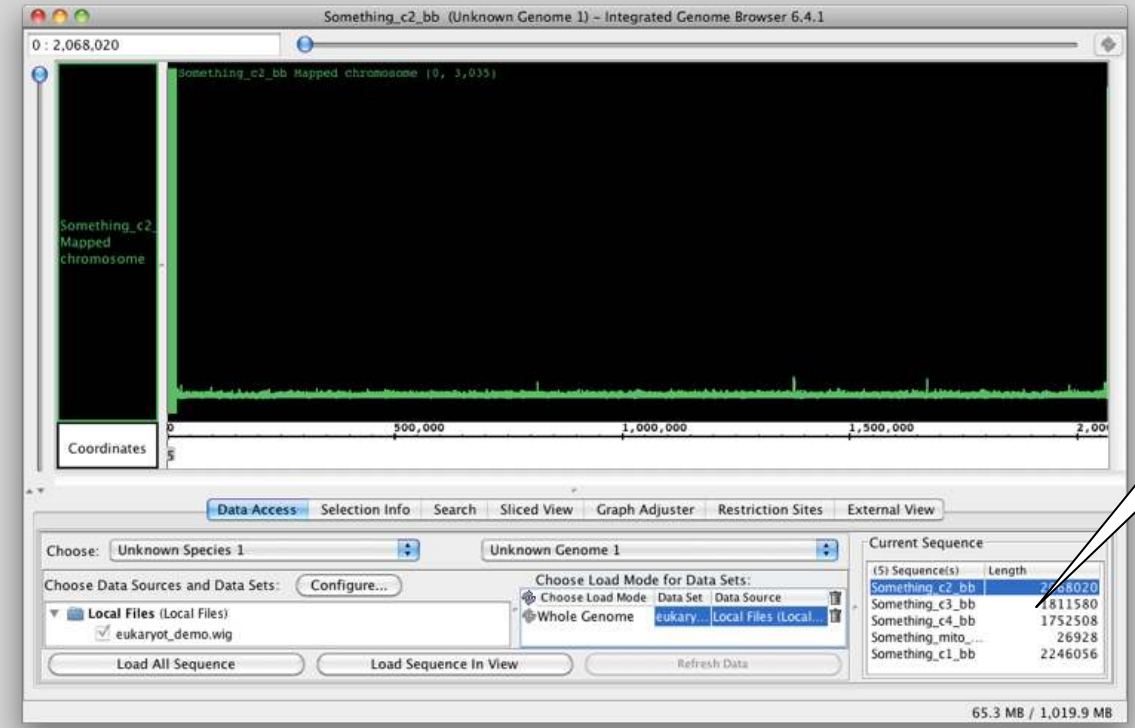
Eukaryotes are mean ... even in mapping assemblies

- close current IGB, restart IGB anew:
  `$ /path/to/IGB/the/staff/tells/you/run_igb.sh`
- navigate to the data folder 07_eukaryot_wig
- load file eukaryot_demo.wig
  - File ➔ Open File
  - do not forget the almost hidden pop-up
    - change "Region in View" to "Whole Genome"



Exercise: 07_eukaryot_wig (3)

Eukaryotes are mean ... even in mapping assemblies



4 chromosomes
(something_c2_bb)

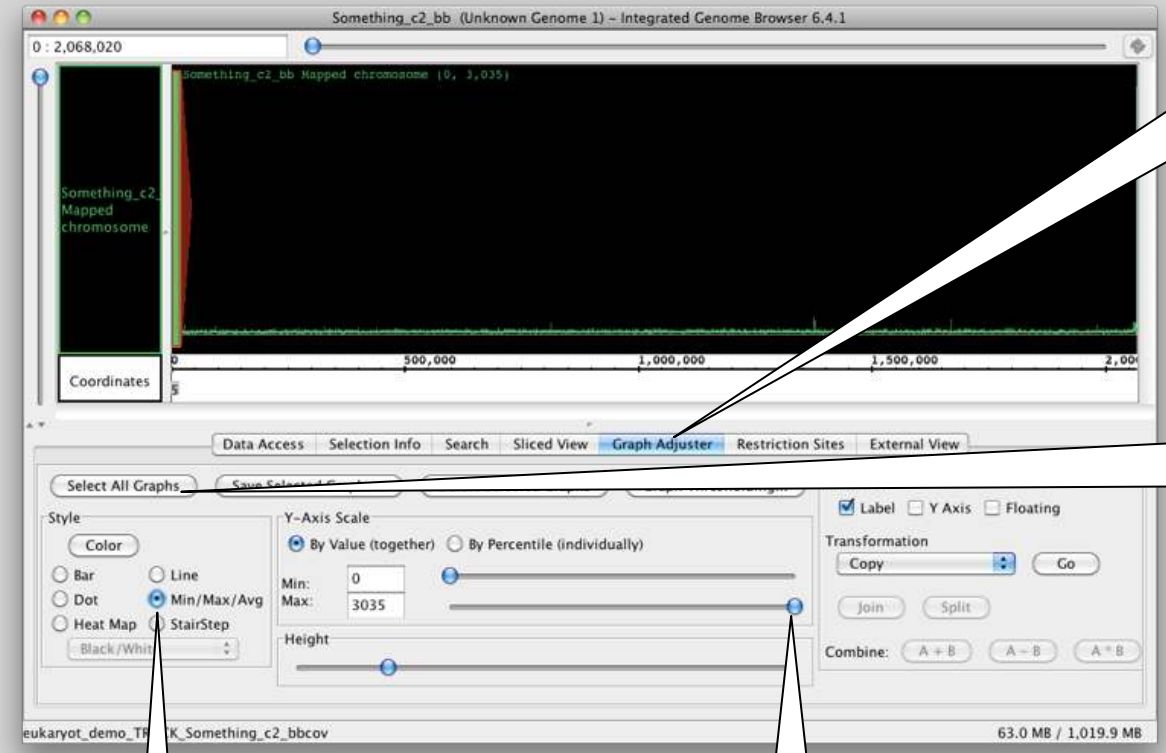
1 mitochondrion
(something_mito_bb)

Before you do anything (click or resize window or whatever ...)

SEE. NEXT. SLIDE!

Exercise: 07_eukaryot_wig (4)

Eukaryotes are mean ... even in mapping assemblies

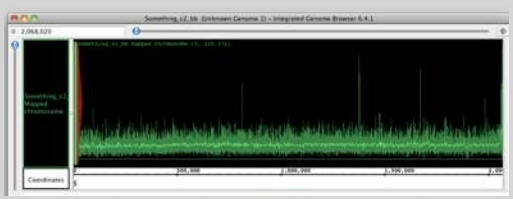


1 Click “Graph Adjuster”

2 Click “Select All Graphs”

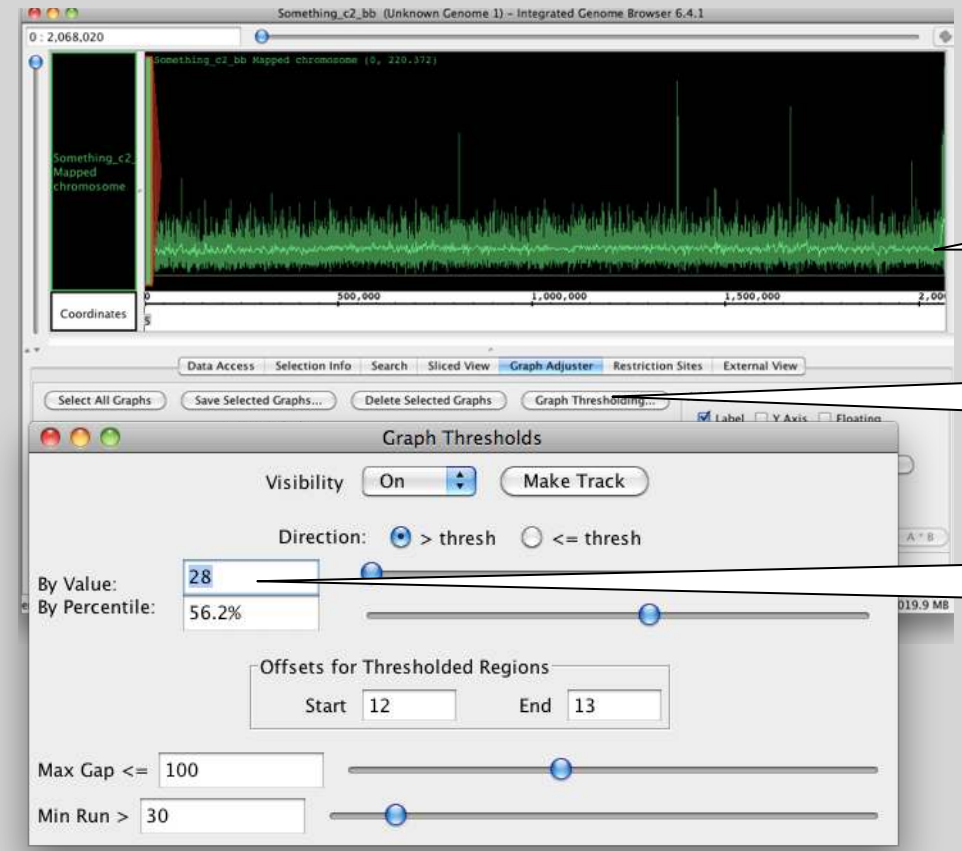
3 Click “Min/Max/Avg”

4 Use the “Max” slider to zoom in a bit, until value is ~220



Exercise: 07_eukaryot_wig (5)

Eukaryotes are mean ... even in mapping assemblies



3 See a line appear here.

Approximately covering the light green “Avg”

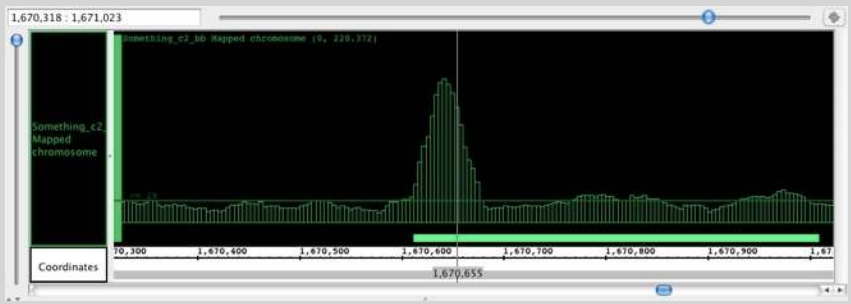
1 Click “Graph Thresholding”

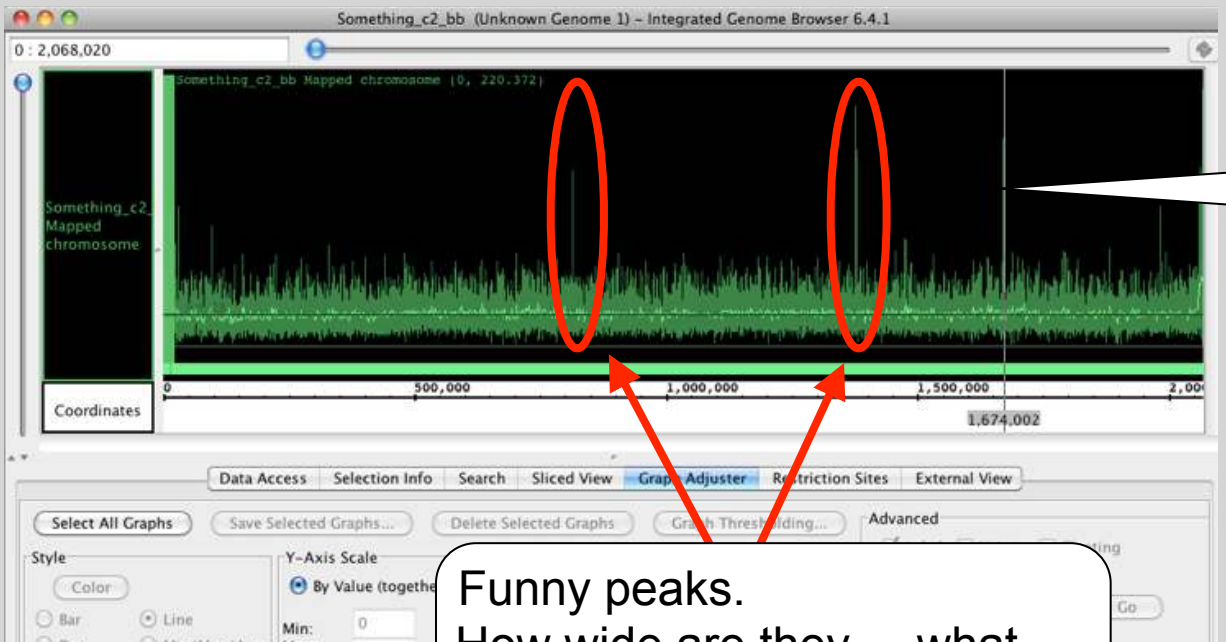
2 Enter “28” in the “By Value” text box (or use the slider to the right)

Exercise: 07_eukaryot_wig (6)

Eukaryotes are mean ... even in mapping assemblies

2 Use the Zoom slider to zoom in





1 Set the “cursor bar” on that funny peak (just click there)

Does not need to be exact, can be reset during zoom process

Funny peaks.
How wide are they ... what could they be?

Exercise: 07_eukaryot_wig (7)
Eukaryotes are mean ... even in mapping assemblies

- Exercise: Find one large scale genome duplication
  - Name chromosome and approximate positions
  - Remember: use “Min/Max/Avg” to work fluidly

- Find out what the funny peaks may be
  - give a biological/technical explanation
  - Remember: look at all contigs (4 chromosomes and mitochondrion), comparing their properties

Exercise/demo: 04_bs168_keio_reseq (1)

Having fun: resequencing a simple bacterium

- Bacillus subtillis 168
  - model organism, 4.2 mb
  - published 1997 (≥ 2000 errors)
  - re-published 2008 (454 & Solexa, ≈40 errors)
- Data set
  - BS168 resequenced by Keio University
  - 35bp, paired-end

Bigger data set, demo only.

NCBI Short Read Archive (SRA)

- stores public data from 454, Solexa, SOLiD, etc.pp
  - cousin of the Trace Archive at NCBI which stores Sanger data
- new development: data stored in own SRA format
  - good
    - “standardized” format, tools to convert to FASTA, FASTQ, etc.pp
    - pre-processed, no need to do tricky things yourself (e.g. splitting 454 paired-end etc.)
  - bad
    - pre-processed data means risk of wrong pre-processing
    - no storage of original data ... no second chance of processing!

Exercise/demo: 04_bs168_keio_reseq (3)

Having fun: resequencing a simple bacterium

- Basically, same workflow as exercise 02
  - map Solexa reads (only the _1 data set)
  - search SNPs, eventually clean up in gap4
  - use convert_project to do SNP analysis
  - Somewhat bigger difference: Solexa reads merged!

- Further possibilities
  - map data sets of both pairs ("_1" and "_2"), without merging and do analysis of insert sizes

Exercise/demo: 05_bsnatto_keio_reseq (1)

Having less fun: resequencing when reference is further away

- Bacillus subtilis natto
  - never sequenced / published until recently
  - “close” relative to BS168
  - used for production of Natto:
    - traditional Japanese food made from soybeans fermented with Bacillus subtilis

- Data set
  - BS168 resequenced by Keio University
  - 35bp, paired-end

Bigger data set, demo only.

- Acinetobacter ADP1
  - 454 and Solexa data sets available at the NCBI SRA
  - downloaded 2009 (2008?) from the SRA in original formats (FASTQ for Solexa, SFF for 454)
  - showing different options, strategies, problems in these data sets.

Bigger data set, demo only.

Exercise/demo: 06_aciadp_baddata (1)
Trust no one

- Acinetobacter ADP1
  - same data set as exercise/demo 04!
  - only 454 data this time
  - downloaded December 2010 as in SRA format
  - the trouble with 454 paired-end linker ...

Bigger data set, demo only.

MIRA parameters

Diff between different --job runs

The End

Thank you.