Perjantai 11.1. : Assembly, Scaffolding.

- Connect to hippu.csc.fi using PuTTY, enable Xming.

Set a name to a path to the work folders prepared by the teacher:

```
set COURSEDIR=/fs/lustre/wrk/somervuo/ngskurssi

set DATDIR=$COURSEDIR/data

set REF=$DATDIR/NC_000913.fna

set SCRIPTS=$COURSEDIR/scripts
```

Data: Illumina PE, 600x E.coli (SRA: ERP000092).

```
zcat $DATDIR/ERR022075pe.fasta.gz | head -1500000 > subset.fasta
```

- zcat uncompresses on the command line and concatenates files and print on the standard output
- head -1500000 shows first 1500000 bytes of the file.

Data quality checking:

```
module load fastx

zcat $DATDIR/ERR022075_1.fastq.gz | head -1500000 > reads1.fq

zcat $DATDIR/ERR022075_2.fastq.gz | head -1500000 > reads2.fq

fastx_quality_stats -i reads1.fq -o reads1.qv
```

- *fastx* –i input –o output. -> txt-file with quality scores on different lines.
- The quality report contains information such as the minimum, maximum, mean, median and inter-quartile range of quality values in each position (cycle) of the reads. Also the number of positions (the length of reads) and the number of A, T, C and G in each position is listed

Task: How average quality goes towards the end of reads?

- The average quality drops towards the 3'-end. Incorrect bases negatively impact assembly, mapping, and other downstream analysis.

Genome de novo assembly with Velvet:

Velvet is an assembler designed for short read sequencing. Velvet currently takes in short read sequences, removes errors then produces high quality unique contigs. It then uses paired-end read and long read information, when available, to retrieve the repeated areas between contigs.

k-mers: The sequence split into units of length k. These k-mers are entered into a hash table and contigs are constructed using a deBrujn graph.

When choosing k-mers for Velvet:

- Number must be odd.
- It must be below or equal to MAXKMERLENGHT variable in the global.h file.
- It must be strictly inferior to read length.

Example 1: Velvet with single read information.

```
module load velvet

velveth vdirS 23 -fasta -short subset.fasta

velvetg vdirS -exp_cov 15 -min_contig_lgth 1000
```

Velveth reads in these sequence files and produces a hashtable, and populates the hashtable with k-mers of length 23, and makes two output files (Roadmaps and Sequences).

Velvetg is the core of Velvet where the de Bruijn graph is built then manipulated. It's ran with the expected coverage of 15 and min. contig length is allowed to be 1000nt. When velvetg finishes it outputs the number of nodes, n50 (average contig length), and max and total size of the assembly created. In the auto_* directory, you will also see a few files: contigs.fa, LastGraph, PreGraph, Sequences, Graph2, Log Roadmaps, stats.txt .

```
Output: Final graph has 12344 nodes and n50 of 3405, max 13878, total
4677483, using 684519/750000 reads
```

```
$SCRIPTS/fastalen.pl vdirS/contigs.fa > contigsS.len

$SCRIPTS/n50.pl contigsS.len

1345 sequences, sum_length: 4187986
min_length: 1000, max_length: 13900, N50: 3776
```

Perl script fastalen.pl calculates contig lengths, it takes the first line in the output.fa as header and then calculates the length of the following sequence. n50.pl calculates n50 value of summary of contig lengths.

congtigS.len example line:

```
>NODE_1_length_1888_cov_14.120233       1910
```

Example 2: Velvet assembly with paired end information.

```
velveth vdirP 23 -fasta -shortPaired subset.fasta

velvetg vdirP -exp_cov 15 -ins_length 500 -min_contig_lgth 1000
```

Velvet only handles interleaved or “shuffled” fasta and fastq files, where each pair is seen one after the other. Our data was already in this format, but velvet has a script to do it too.

```
shuffleSequences_fastq.pl input1 input2 > output
```

Output: `Final graph has 10227 nodes and n50 of 57746, max 199412, total 4747681, using 723696/750000 reads`

```
$SCRIPTS/fastalen.pl vdirP/contigs.fa > contigsP.len

$SCRIPTS/n50.pl contigsP.len

139 sequences, sum_length: 4496757
min_length: 1028, max_length: 199434, N50: 59708
```

If you only want to know how many lines were in the file:

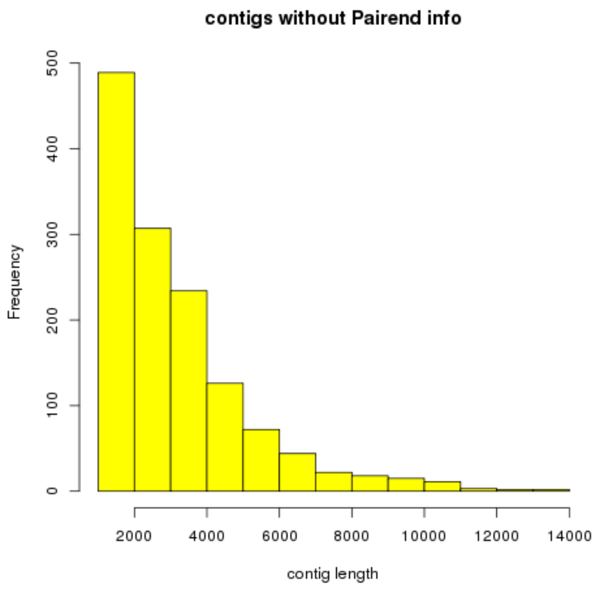
```
wc -l contigsP.len
```

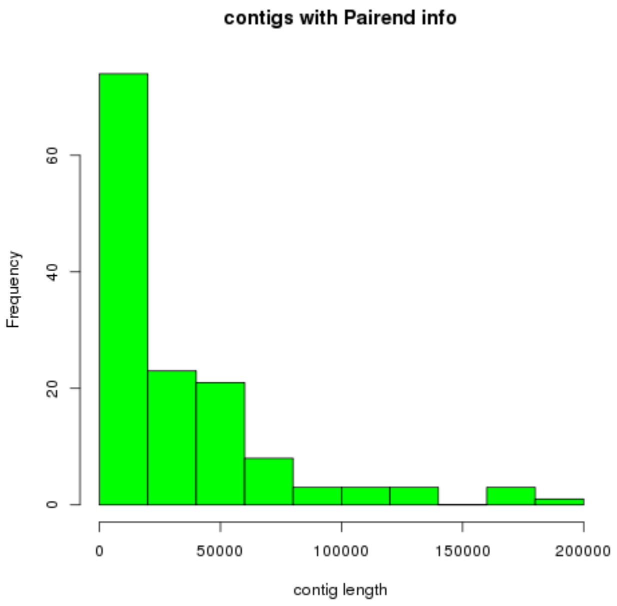
Visualisation:

We can plot contig lengths e.g. using R which is a software environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The R environment is a command line “language”.

"contigsS.len" and "contigsP.len" must be available in working directory of R.

```
module load R
R
a=read.table("contigsS.len", header=F)
b=read.table("contigsP.len", header=F)
hist(a[,2],col="yellow",main="contigs without Pairend info",xlab="contig
length")
x11()
hist(b[,2],col="green",main="contigs with Pairend info",xlab="contig
length")
d=b[,2]
x=sort(d,dec=T)
cx=cumsum(x)
plot(x,main="sorted contig lengths")
dev.set(2)
plot(cx,main="cumulative contig lengths")
v=sum(d)/2
abline(h=v,col="red",lwd=3)
i50=min(which(cx>v))
# this is N50:
x[i50]
dev.set(3)
abline(v=i50,col="blue",lwd=3)
```

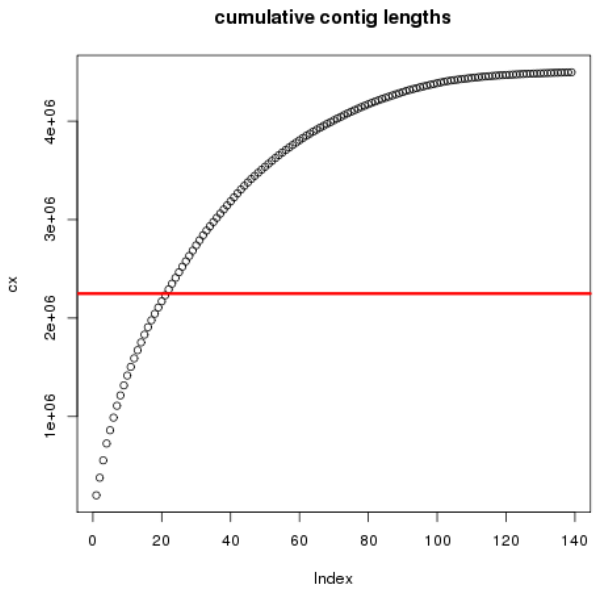


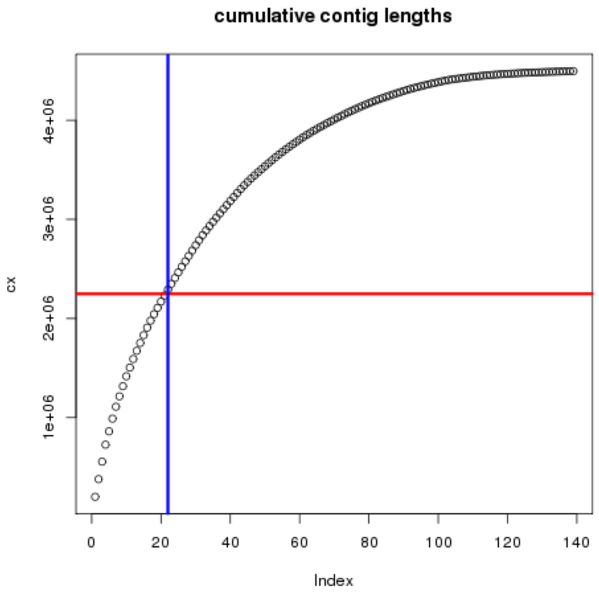


Histograms for contigsS (singles) and contigsP (paired).

Contigs on paired info are on average much longer than contigs without.

In paired end data (contigsP):





Red line is average, blue line is N50. N50 is the length of shortest contig which corresponds to half of the total sum, i.e., half of the genome size, when summing sorted contig lengths. So it will intersect the average where the contigs intersect it.

Compare assemblies with and without pairend read information (number & length of contigs, N50).

- Paired end run produces less contigs that are longer. There were 139 contigs in paired and 1345 contigs in single. n50 values of paired run were much higher (57746) than the single run (3405). This means paired end information produces better quality contigs.

How the choice of kmer length affects the assembly?

- It's ia trade-off between specificity and sensitivity. Longer kmers bring you more specificity and velvet will be much slower to run.

How parameters exp_cov and cov_cutoff affect the assembly?

- velvet doesn't know if a read it has is unique (f.ex. contains SNP and is hard to align) or from a repetitive region. With expected coverage you can guide velvet to make good guesses as to what the reads are. With low exp_cov, velvet will dismiss more reads as repetitive and tries to assemble all of them separately. If exp_cov is high it will assume they all come from the same genomic region and will take weighted average of the reads to build up the region.
- cov_cutoff removes low coverage nodes during assembly.
- both can be run as “auto”, so velvet will determine the required values.

Can you think reasons why there are gaps in assembly, i.e. why we don't get a single long contig representing the entire genome even when using high sequencing depth? How we could improve assembly?

- Low coverage, bad read quality (sequencing errors, other technical errors), repetitive genomic sequences, reads deteriorate from both ends, methods are heuristic and very memory intensive: final answer is not always absolutely true.
- Longer reads, high coverage, minimize errors.

Example 3: assembly with 60X coverage data:

```
zcat $DATDIR/ERR022075pe.fasta.gz | head -5400000 > datape.fasta

velveth vdirP23 23 -fasta -shortPaired datape.fasta

velvetg vdirP23 -ins_length 500 -exp_cov 60 -min_contig_lgth 1000 -
scaffolding no
```

Final graph has 51583 nodes and n50 of 40309, max 152847, total 5455686, using 2596925/2700000 reads

```
$SCRIPTS/fastalen.pl vdirP23/contigs.fa > len1

$SCRIPTS/n50.pl len1

150 sequences, sum_length: 4518249
min_length: 1011, max_length: 152869, N50: 47582
```

Sort contig names based on coverage and second is length:

```
grep '^>' vdirP23/contigs.fa | sort -n -k6 -t"_"

grep '^>' vdirP23/contigs.fa | sort -n -k4 -t"_"

The line looks like
```

>NODE_16828_length_97600_cov_41.902645

Grep pics up every line that starts with “>” and then sort splits between “_”’s. The fourth field when split by “_” is length (-k 4) and sixth is coverage (-k 6). It can then sort these reads based on those fields.

Assembly validation:

To determine how good our contigs are we can map them against the genome. We use bwa:

```
bwa index -a is $REF

bwa bwasw -t 4 $REF vdirP23/contigs.fa > contigs.sam
```

BWA is a software package for mapping low-divergent sequences against a large reference genome. We use the bwa-sw algorithm, that’s designed for long sequences (70bp-1Mbp). bwa index indexes database sequences in the fasta format. Then the alignment algorithm bwasw is run, with –t for multithreading with 4 threads on the hippu-server. It creates a SAM-file, that contains all the information on where the reads aligned and how well they aligned.

One line:

```
@SQ       SN:gi|49175990|ref|NC_000913.2|        LN:4639675
NODE_38_length_1246_cov_14.805779       16
          gi|49175990|ref|NC_000913.2|  734360    250       1268M     *
          0
          aTGGAATTTTTATCAGTATCCG(…)GAGATACTACTTATTGTCTAAAGCTGTTCATAGCTT
          *         AS:i:1268 XS:i:0    XF:i:0    XE:i:26   NM:i:0
```

Fields are tab separated and first line with @SQ is the header line for the whole file, containing information relative to all reads. This file could have @RG information about Read Groups from the sequencing run. Then one line of the contigs we had, contig name (NODE_38 (…)). SAM-flag (tells things like if both reads in the pair are mapped), reference name, where it mapped on the reference (position), mapping quality. position of pair… etc. The last things on the line XS:i:0 are tags from bwa. They tell about the quality of the alignment of the read. f.ex. XT:A:U means the read was uniquely mapped.

Where contigs mapped was the visualized with tablet and hawkeye.

Assembly with SOAPdenovo

SOAPdenovo is a novel short-read assembly method that can build a de novo draft assembly for the human-sized genomes. The program is specially designed to assemble Illumina GA short reads.

Task 1. Create config file soap.config with text editor. I used “nano”. This information is what you have about your sequencing run, like read length, insert sizes, etc. You can set cutoffs to ignore the ends of reads.

Running SOAPdenovo: you set it with –s. K-mer is set with –k and –p is number of cpu (threads) for use.

```
module load soapdenovo
SOAPdenovo-31mer all -s soap.config -K 23 -o soap23b -p 4

$SCRIPTS/fastalen.pl soap23b.contig | $SCRIPTS/n50.pl
13900 sequences, sum_length: 5285746
min_length: 24, max_length: 18659, N50: 2683

$SCRIPTS/fastalen.pl soap23b.scafSeq | $SCRIPTS/n50.pl
180 sequences, sum_length: 4552736
min_length: 100, max_length: 221391, N50: 130372
```

Results from different cutoff runs (50,75):

```
$SCRIPTS/fastalen.pl soap2.scafSeq | $SCRIPTS/n50.pl
117 sequences, sum_length: 4598638
min_length: 100, max_length: 1090560, N50: 651708

SCRIPTS/fastalen.pl soap2.contig | $SCRIPTS/n50.pl
13900 sequences, sum_length: 5285746
min_length: 24, max_length: 18659, N50: 2683

$SCRIPTS/fastalen.pl soap3.scafSeq | $SCRIPTS/n50.pl
100 sequences, sum_length: 4659289
min_length: 100, max_length: 4636497, N50: 4636497

$SCRIPTS/fastalen.pl soap3.contig | $SCRIPTS/n50.pl
13900 sequences, sum_length: 5285746
min_length: 24, max_length: 18659, N50: 2683
```

The higher the cutoff the longer the sequences and the larger the N50.

Perjantai 18.1.

Lecture notes:

Gene prediction, a pattern recognition problem

- Ab Initio (Itseoppivat: etsii geenisignaaleja)
  - Prokaryootit: esim. S-D, -10/-35. Alku(Met) ja lopetuskodoni samassa lukukehyksessä. Ei eksoni-/intronirakennetta, esim. S-D on aina läsnä. -> helpompaa.
  - Eukaryootit: 3% koodaustiheys, eksoni-/introni-rakenne, isot genomit, kaikki signaalit (kuten Kozak) eivät joka geenissä -> vaikeuttaa. Geenisignaalit: esim. RNAPolII prom. elementit, Kozak (RBS), PolyA, ..
    - Sisältösignaalit:
    - GC% korkeampi koodaavassa alueessa (45-50%), mutta ei riitä yksinään koska myös CpG-saarekkeet ovat GC tiheitä. Pelkän GC-perusteella ~63% oikein.
    - ”Codon Usage Bias”. Jotkut kodonit käytetympiä koodaavalla alueella ja joitakin ei esiinny juuri ollenkaan. Voidaan laskea todennäköisyys sille, että kyseinen sekvenssi ei ole koodaava alue ja sen log-likelyhood. (- => ei ole koodaava, + => koodaava) Pelkkä kodon-bias ennustustarkkuus ~87%.
    - Splice-sitet. 5’ GU (Donor), 3’ AG (Acceptor). [Exot1]GU ----AG[Exon2]. Vain ~0.5%:n ei päde.
  - Bakteeridatan elementit tunnetaan hyvin, mutta eukaryoottidata on niin vaikeaa, että kehitys hitaampaa.
  - HMM: Tarvitaan kaksi opetusjoukkoa CpG-aluetta ja ei-CpG-aluetta. Muodostetaan +/- matriisi, jossa siirtymätodennäköisyydet. Kysytään mitä mallia sekvenssi noudattaa? Jos siirtymät vastaavat paremmin +-matriisin arvoja -> koodaava.

        - GHMM: Monia tiloja joiden on toteuduttava ja ovat luonteeltaan sellaisia, että kun kaavion läpi käydään ”oikein” järjestyksessä niiden toteutuminen varmentaa onko kyseessä e. eksoni/introni.
    - Olemassaoleva data, e. RNASeq. Ei voi löytää geenejä, joita ei ekspressoida.
    - Paras käyttää molempia menetelmiä.

Set correct parths for programs prodigal, RNAmmer, tRNAscan, Genemark. Then get the genome with wget. Then check if there are Windows linebreaks in file with the perl script, because those will cause an error in the next step.

```
perl -pe '$_ =~ s/\r//' SCC3193.fasta > SCC3193_no_breaks.fasta
```

To check if anything was removed using word count:

```
wc SCC3193.fasta
73763   73775 5312072 SCC3193.fasta
wc SCC3193_no_breaks.fasta
73763   73775 5238309 SCC3193_no_breaks.fasta
```

The linebreaks were removed! I added a “>” on the header line of the .fasta file so that it’s in line with the fasta format or Prodigal wouldn’t run.

Predict genes: Run Prodigal:

```
prodigal -a SCC3193.pep -i SCC3193_no_breaks.fasta -f gff -o SCC3193.gff3
```

-a makes a amino acid sequence. –f makes the file format into gff.

The aminoacid file “SCC3193.pep” has the sequences converted to amino acids. The GFF3 file, headers are commented out with #.

```
##gff-version  3
# Sequence Data: seqnum=1;seqlen=5164464;seqhdr="Pectobacterium sp. SCC3193,
complete genome - "
# Model Data: version=Prodigal.v2.60;run_type=Single;model="Ab
initio";gc_cont=50.37;transl_table=11;uses_sd=1
Pectobacterium  Prodigal_v2.60  CDS     3       1451    174.8
   +       0       ID=1_1;partial=10;start_type=Edge;rbs_motif=None;rbs_spacer=
None;gc_cont=0.521;conf=100.00;score=174.77;cscore=171.55;sscore=3.22;rscore=0.
00;uscore=0.00;tscore=3.22;
```

All the lines have one read and it’s score and other information. The information on the line is tab separated.

Predict tRNAs:

tRNAscan-SE was designed to make rapid, sensitive searches of genomic sequence feasible using the selectivity of the Cove analysis package.

```
tRNAscan-SE -B -o SCC3193_tRNA.out SCC3193_no_breaks.fasta
```

-o *file*

: save final results in *file*

Specifiy this option to write results to *file* rather than standard output.

-B *number*

: use *number* nucleotides padding for first-pass tRNA predictions

By default, tRNAscan-SE adds 7 nucleotides to both ends of tRNA predictions when first-pass tRNA predictions are passed to covariance model (CM) analysis. CM analysis generally trims these bounds back down, but on occassion, allows prediction of an otherwise truncated first-pass tRNA prediction.

tRNAscan doesn't support gff format in output, so we need to do file format conversion separately.

```
wget http://ekhidna.biocenter.helsinki.fi/downloads/teaching/spring2013/conve
rt_trna_to_gff.pl

perl convert_trna_to_gff.pl SCC3193_tRNA.out > SCC_tRNA.gff2
```

We would like to predict all different rRNA molecules (5/8s, 16/18s and 23/28s) from *P. wasabiae* SCC3193 genome and we would like to have results in gff format. Scan both strands at the same time. What parameters we should use in this case?

```
rnammer -S bac -gff SCC3193_rrna.gff2 -multi -m tsu,ssu,lsu
SCC3193_no_breaks.fasta
```

-multi Runs all molecules and both strands in parallel

-m Molecule type can be 'tsu' for 5/8s rRNA, 'ssu' for 16/18s rRNA, 'lsu' for 23/28s rRNA or any combination seperated by comma.

-gff output gff file
Specifies filename for output in GFF version 2 output

-S Specifies the super kingdom of the input sequence. Can be either 'arc', 'bac', or 'euk'.

ARGO:

Bacteria Genetic Code: 11

Count SSC_tRNA.gff2 77

Chromosome Statistics

Length (bps)	27,905,053
Protein coding gene count	3,370
Non coding gene count	239
Pseudogene count	26
Variations	1,397,489

GeneMark ES:

```
cp /p/appl/molbio/course/GeneMark/gm_key_64_es.tar ~/
tar -xvf ~/gm_key_64_es.tar -C ~/
mv ~/gm_key ~/.gm_key
perl /p/appl/molbio/course/GeneMark/gmes/gm_es.pl
Drosophila_melanogaster.BDGP5.69.dna_rm.chromosome.3R.fa
```

The HMM architecture of eukaryotic GeneMark.hmm consists of hidden states for initial, internal and terminal exons, introns, intergenic regions and single exon genes located on both DNA strands. It also includes hidden states for initiation site, termination site, as well as donor and acceptor splice sites. GeneMark.hmm has been frequently used for annotation of plant and animal genomes.

The GeneMark-ES run never completed, I tried leaving it running on hippo to check when I get back home, but it had stopped.

Friday 25.1. : Genome structural annotation (Apollo)

Lecture notes:

Annotation

Repeats: Low-complexity Repeats (poly-purine, high AT/GC, tandem repeats), Retro-/DNA-transposons. Pseudogenes/multicopy genes...

Known repeats can be masked (soft/hard) to avoid low-complexity areas to peak/mask under important reads.

- Repbase database for known repeats

De novo repeat prediction: calculate consensus representing repetitive DNA from study species.

Pairwise alignment of the genome -> cluster alignments and find/define elements to align again, group substrings into repeat families.

RepeatScout calculates consensus local alignment Q (Mapping Quality?) for large fragments

-> fit alignment for smaller fragments -> fit-preferred alignment score for medium sized fragments

Finds a window in which to search for similar repeats with maximum score.

RepeatMasker

Remove repeats from the genome by masked by the RepeatScout information and already known repeats from similar species etc.

Esim. Repeat masking of M.cinxia genome

Pelkällä RepBasella vain ~7.5%, lisäksi RepeatScout/Recon/Piler… -> ~40% contigeista, ~32% scaffoldeista.

Gene prediction with MAKER

ab initio -> matemaattinen malli, hmm ja opetusjoukko esim. sellaisia eksoneita, jotka tunnetaan varmasti -> niiden perusteella lasketaan todennäköisyysmatriisi (e. miten usein yhden eksonin geeni /kahden/etc muita tapahtumia.) Opetusjoukko tärkeä.

With MAKER -> käytetään tunnettua tietoa databaseista (proteiinit etc) etc. Maskataan pois tunnetut ja ennustetut repeatit, RNA-Seq assembly/mapping.

MAKER maskaa high-complexity hardilla ja low-complexity softilla.

MAKER ab initio gene prediction with SNAP.

SNAP uudelleeniteroi opetusjoukkoaan

Protein evidence: Uniprot tarkistetut (refseq) sekvenssit läheisistä suvuista, tai konservoituneet proteiinit

BlastX löytää konservoituneiden proteiinien sekvenssit / Exonerate pakottaa splice/sitet oikein (gt….ag erittäin konservoitunut signaali) .

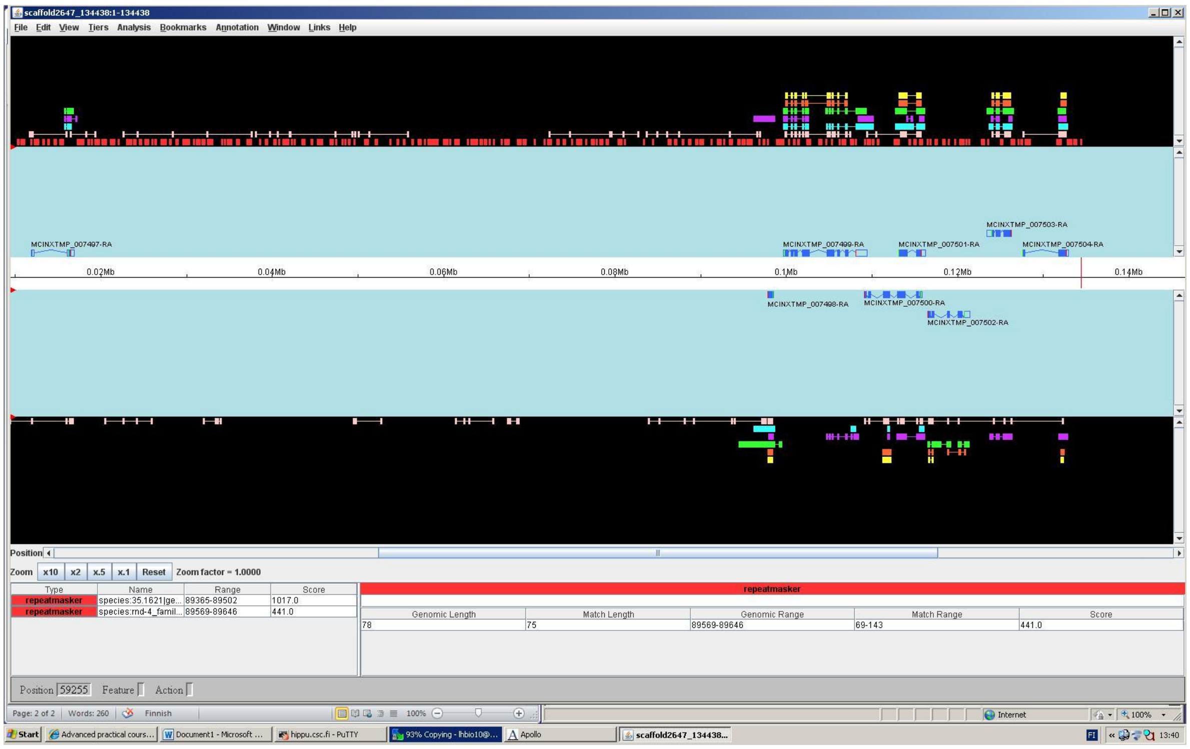
Exercises:

Open scaffold2647_134438.gff in wordpad. What does the file contain ?

- File contains everything what maker has done: standard orientation, frames, exon, all info from SNAP, RepeatMasker, etc. Everything is in gff3 format.

Open scaffold2647_134438.gff in gff3 format in Apollo. Change the colors for tiers and save them to type preferences.

- All different types of results are in different color (f.ex.protein2genome is blue). Boxes are exons, lines introns.



Type	Name	Range	Score
repeatmasker	species:35.1621|ge...	89365-89502	1017.0
repeatmasker	species:rnd-4_famil...	89569-89646	441.0

repeatmasker

Genomic Length	Match Length	Genomic Range	Match Range	Score
78	75	89569-89646	69-143	441.0

Position 59255 Feature Action

View in Apollo. The middle is the gene predictions and on top and bottom are the strands and the predictions by type, all with different color. You can zoom in using the tools in the bottom of the page.

Check and correct, if necessary, the gene model MCINXTMP_007504-RA in Apollo. Save the corrected model in gff3 format.

- After altering the exons a bit according to blastx and blastn results and ignoring cufflinks only results I BLASTed the sequence agains nr:

- 1 ribosomal protein S5 [Manduca sexta] 446 446 98% 1e-157 99% ACY95347.1
- 2 ribosomal protein S5 [Trichoplusia ni] 442 442 98% 8e-156 98% AAX63830.1
- 3 ribosomal protein S5 [Euphydryas aurinia] 441 441 98% 1e-155 97% ADT80675.

Score	Expect	Method	Identities	Positives	Gaps	Frame
446 bits(1147)	1e-157()	Compositional matrix adjust.	216/219(99%)	216/219(98%)	1/219(0%)	

The protein is correct and I have edited it very well (nearly full identity). I also made a MAFFT-alignment with this and other similar proteins to confirm.

Open scaffold825_259930.gff and check the model MCINXTMP_015555-RA. Check the predicted protein similarly than in 6).

I removed two exons from the start of the predicted gene, because they overlapped with another gene and they had no other supporting data other than SNAP. Results from blastx etc. started after two exons. These two exons were very far away from the rest of the predicted gene.

I made a new starting point by expanding according to est2genome and est_gff_cufflings and made the first exon a bit longer than predicted, because the suggested starting codon would have not made a complete protein.

All exons except the last one (with stop) looked very good and supported by most data. The last exon was unclear/strange and it wasn't supported by blast results, but was supported by SNAP. I left it there, because it seemed okay otherwise and stop codon was good.

Blast results against uniprot:

1 RecName: Full=60S ribosomal protein L13a 400 400 100% 1e-141 94% Q8MUR4.1

2 RecName: Full=60S ribosomal protein L13a 391 391 100% 4e-138 92% Q962U0.1

3 RecName: Full=60S ribosomal protein L13a 289 289 100% 6e-98 65% Q9VNE9.1

All of the best results are butterflies, similarity goes down when results are other insects.

Score	Expect	Method	Identities	Positives	Gaps	Frame
400 bits(1028)	1e-141()	Compositional matrix adjust.	191/204(94%)	200/204(98%)	0/204(0%)	

Very similar to the best result, good edit.

Check and correct the gene model MCINXTMP_015558-RA from scaffold825_259930. Use blast and Ensembl sequence search for checking the predicted protein.

I rearranged the middle exons length because it wasn't according to any of the "known" results, like blasts. I also realigned the last exon to be more according to blast and SNAP results.

BLAST results produce a Serendipity A-protein family protein.

Results from Ensembl Metazoa:

Heliconius melpomene	HE670847	169	495	65 %	4E-53	View	HMEL008918

The prediction was okay, I've probably made some slight errors in the middle codon when I altered it. The alignment was mostly erroneous in the middle.

Check and correct the gene model MCINXTMP_007503-RA from scaffold2647_134438

I changed the length of the last two exons so that the splicesite is correct between them and they are more in line with blast/protein2genome results.

I then BLASTed the sequence and the result was a NADPH:adrenodoxin oxidoreductase, mitochondrial.

Score	Expect	Method	Identities	Positives	Gaps	Frame
444 bits(1143)	5e-151()	Compositional matrix adjust.	232/446(52%)	297/446(66%)	21/446(4%)	

Identity was not perfect, The alignment shows most of the inconsistencies in the end of the protein. Everyone in class got the same result.

Friday 1.2: Genome structural annotation (Apollo)

Set paths and load all needed programs and files, like before. This time also using bioperl, the Perl package with biologically oriented programming tools.

Run RepeatMasker:

RepeatMasker is a program that screens DNA sequences for interspersed repeats and low complexity DNA sequences. The output of the program is a detailed annotation of the repeats that are present in the query sequence. On average, almost 50% of a human genomic DNA sequence currently will be masked by the program. [Source: Repeat Masker documentation]

```
RepeatMasker -dir repeats_RepBase -species all scaffold3244.fasta

RepeatMasker -dir repeats_denovo -lib denovo_repeats.fasta scaffold3244.fasta
```

Results:

.masked file that has hard masked repeats (areas with repeats are replaced with N's).

.tbl file that says where and how many repeats have been masked.

- Multiple runs find slightly different amounts of repeats, so as results there's slightly differently masked end files.
- (all) Total interspersed repeats: 9751 bp 1.95 %
    - Unclassified vain ~0.6%.
- (denovo)Total interspersed repeats: 101653 bp 20.31 %
    - Unclassified 18.09% -- Tuntemattomia repeatteja.

Denovo repeat masking is much harder and produces more uncertain results.

Run Maker:

Generate empty control files for Maker: maker –ctl

Edit this control file with an text editor: nano maker_bobts.ctl

Change BLAST-type from wublast to ncbi.

Then give MAaker RNASeq Data and ask it to make gene predictions. Change maker_opts.ctl to say:

```
genome=scaffold3244.fasta
est=RNASeq_trinity.fasta
est_gff=RNASeq_cufflinks.gff
est2genome=1
```

You get the path of the produced files from log or with ls –l. Path in my case was:

`scaffold3244_datastore/B2/EE/scaffold3244_500584/`

I took the .gff file from that directory and trained SNAP with it. Training SNAP means to use known values to readjust the probability values for changing nodes in SNAP.

fathom handles the sequence of max 1kb and sorts it. forge makes estimates and hmm-assembler makes a hmm-model:

```
maker2zff scaffold3244_500584_RNASeq.gff

fathom -categorize 1000 genome.ann genome.dna

fathom -export 1000 -plus uni.ann uni.dna

forge export.ann export.dna

hmm-assembler.pl snap1 . > snap1.hmm
```

Then run maker so that it makes SNAP ab initio predictions using previous data. Edit the maker control files to use the previous .hmm file and the run maker with those settings (snaphmm = snap1.hmm and est2genome=0).

Then we configured new control files so that Maker would use SNAP, RNA-Seq, and protein evidence:

```
est=RNASeq_trinity.fasta
est_gff=RNASeq_cufflinks.gff
protein=scaffold3244_prots.fasta
model_org=all
rmlib=denovo_repeats.fasta
snaphmm=snap1.hmm
```

Open the gff files including the maker gene predictions in Apollo and compare predictions made only using RNASeq data, and all evidence data + SNAP ab initio predictions. What can you conclude ?

In the files produced by the two runs:

snap1_only: no exon predictions, only snap_masked layer.

snap1.gff: many datapoints, very good looking genes.

We also used Augustus for ab initio gene predictions, it had good looking exons.

Conclusions: Ab initio gene prediction is hard, especially with a previously fairly unknown species because there's so little data and then you have to make many predictions based on predictions, and your results might end up not being very reliable and requiring a lot of manual annotation. When you have protein data available the task becomes easier and more reliable.

Friday 8.2. Artemis and RNA-Seq data in manual annotation

Lecture notes:

BAM files are binary SAM-files. They are very compressed, but can be viewed in specialized viewers like Artemis. Artemis can display pairs against the reference (both pairs joined by lines), coverage of the reads, the actual sequence of the reads, etc. You can zoom in to see the read sequnces against the reference and see the constructed consensus. Artemis can also show exons and read frames. To be viewed BAM-files must be indexed. There are many other tools to parse BAMs with. One very good toolset is "bamtools".

Checklist for making gene models in Apollo:

- Check opposite strand
- Start/Stop sites
- Missing/additional exons
- Edges of Exons
- Splice sites
- RNASeq for 5', 3'.
- Pull out SEQ > BLAST > MSA.

Cufflinks

- Very long introns, bad alternative transcripts, merged genes.
- Check Cufflinks data for errors.

Manual annotation:

maker-scaffold3244_500584-augustus-gene-4.4-mRNA-1

I removed the exons that were only in the cufflinks data. I added one exon, that was in all blast*-data and protein2genome-data. I also added one flanking extra exon, that was in RNASeq/Protein2genome data.

BLAST results: putative furrowed [Danaus plexippus]

Alignment statistics for match #1

Score	Expect	Method	Identities	Positives	Gaps	Frame
1759 bits(4556)	0.0()	Compositional matrix adjust.	837/932(90%)	881/932(94%)	1/932(0%)	

augustus_masked-scaffold3244_500584-abinit-gene-3.3-mRNA-1

I removed the beginning of the gene to be according to protein2genome and blast-results. I removed parts from start and end that were only shown in cufflinks and weren't in RNASeq.

BLAST results: hypothetical protein KGM_18655 [Danaus plexippus]

Alignment statistics for match #1

Score	Expect	Method	Identities	Positives	Gaps	Frame
538 bits(1385)	0.0()	Compositional matrix adjust.	264/266(99%)	265/266(99%)	0/266(0%)	

maker-scaffold3244_500584-augustus-gene-2.4-mRNA-1

I removed a very long predicted area that was only based on cufflinks data and was not repeated in RNASeq/prot2genome/Blast... I tried to find a good first exon that had a good start codon and was in all the RNASeq/prot2genomes.

BLAST results: putative receptor tyrosine phosphatase type r2a [Danaus plexippus]

Alignment statistics for match #1

Score	Expect	Method	Identities	Positives	Gaps	Frame
3367 bits(8731)	0.0()	Compositional matrix adjust.	1657/1933(86%)	1710/1933(88%)	177/1933(9%)	

I think I removed a bit too much from the start. In the alignment it shows that I start at base number 144 in the reference, so I've gone over that sequence when I was looking for a good start place.

MCINXTMP_008074-RA from scaffold scaffold2760_48179.gff.

I removed a couple of exons from both the start and the end of the predicted gene, because they had no hits in prot2genome or RNASeq or BLAST. I added a new final exon from RNASeq data.

My first result was not very good:

BLAST result: UDP-glycosyltransferase UGT33F3 [Helicoverpa armigera]

Alignment statistics for match #1

Score	Expect	Method	Identities	Positives	Gaps	Frame
557 bits(1435)	0.0()	Compositional matrix adjust.	263/475(55%)	341/475(71%)	28/475(5%)	

I then elongated one of the middle exons and the last exon:

UDP-glycosyltransferase UGT33F3 [Helicoverpa armigera]

Alignment statistics for match #1

Score	Expect	Method	Identities	Positives	Gaps	Frame
615 bits(1585)	0.0()	Compositional matrix adjust.	284/475(60%)	363/475(76%)	6/475(1%)	

The gaps reduced a lot. Everyone else also had similar results.

Friday 15.2. Functional Annotation

Lecture notes:

DE annotation – Descriptive functional annotation.

GO annotation – Kolme pääluokkaa, hierarkinen rakenne, sisältää tiedon siitä miten tieto on hankittu IEA (Inferred from Electronic Annotation) epäluotettavin, ihmisten tarkistamat luotettavimpia. Malliorganismit usein hyvin annotoituja, mutta esim. bakteerien annotaatiot ovat elektronisesti annotoituja ja epäluotettavempia verrattuna hiivoihin kuten Cancida albicans.

GO Informaatiosisältö: Lasketaan kuinka monta geeniä määrittävät GO-luokan. Sisänode sisältävät aina kaikki siitä polveutuvien lehtien geenit. Jaetaan jokainen lehti ja sisänode juuriarvolla. Otetaan –lg niin saadaan suuri arvo kuvaamaan harvinaista GO-luokkaa ja sitä kuvaavampi se on luonteeltaan. Mitä lähempänä juurta sitä abstraktimpi GO on.

Voidaan laskea semanttisia etäisyyksiä kertomaan kuinka samanlaisia GO-luokat ovat keskenään.

Resnik = yhteinen ”esi-isä”.

Lin = Yhteinen vanhempi kerrotaan kahdella ja jaetaan kahden ontologian informaatiosisällön summalla.

Korreloiko semattinen etäisyys sekvenssisamanlaisuuden kanssa? Riippuen laskentatavasta ja GO-luokasta. Esim. Resnik toimii parhaiten Molecular Function-yläluokassa, Lin toimii pairhaiten Cellular Component –luokassa, jne.

EC annotation – Enzyme Comission number . Proteiinien entsyymiaktiivisuus.

Virheiden rikastuminen tietokannoissa:

Tarkista aina: Linjauksen kattavuus (huom. BLAST-kattavuus vain yhdensuuntainen).

Weakening Signal

- Manuaalisesti kuratoitu proteiini (oikea) lisätty tietokantaan, sitä kohti liitetään elektronisesti annotoituja sekvenssejä, joita kohti lisätään… jne. Etäisyys viimeisestä ”oikeasta” annotaatiosta kasvaa ja funktionaalisuudet eivät enää ehkä vastaa toisiaan.

Domain Walking

- Yhden domainin proteiini käytetään annotoimaan monen domainin proteiinia, kun monen domainin proteiinia kohtaa linjataan lisäproteiini se saa myös väärän funktionaalisuuden.

Funktion määrittäminen: Sekvenssihomologia:

GOTCHA:

BLAST haku -> Tehdään GO luokkia BLAST tuloksista -> -log(E-value) käytetään BLASTin pisteytykseen -> Yhdistä tulokset geeneistä, jotka kuuluvat samaan GO-luokkaan (puskee tietoa ylemmäs puussa)

- Ei aina spesifejä tuloksia, koska tulos joka oikeasti kuuluisi löytää lehdestä (spesifinen toiminnallisuus) saattaa löytyä hyvin yhäältä rootin läheltä, koska myös kaikki ylemmät luokat sisältävät kaiken funktionaalisuuden alemmista luokista.

Argot:

Korjaa GOTCHA:n ongelmia antamalla myös painotusarvon informaatiosisällölle (kuinka pitkällä ollaan puussa). Clusteroi samanlaisia geeniontologioita.

PANNZER:

Lisää taksonomisen etäisyyden (NCBI-taksonomiapuu), k-nearest,

Vähemmän menestyvät metodit ennustavat suuria GO-luokkia, pieni informaatiosisältö. Hyvin menestyvät metodit: JonesUCL, Argot2, PANNZER.

Exercizes:

FigFAM scan example:

Query Sequence	>ECOLI_0 …
Hit Family	FIG000885
Hit Family Function	Aspartokinase (EC 2.7.2.4) / Homoserine dehydrogenase (EC 1.1.1.3)

Run batch job with RAST server using Perl script that automates the annotation

calls:

```
perl /fs/appl/hippu/users/jpkoskin/RAST/anno_call.pl <
P_wasabiae_SCC3193.fasta > 250_output
```

output →

```
W5S_0001      Chromosomal replication initiator protein DnaA      55
…
```

W5S_0250	Cystine-binding periplasmic protein precursor	Serratia marcescens Db11	17

GO Annotation:

Using the p53 protein sequence, try different methods to functionally annotate it.

GOTCHA: It stays on a very basic level, it does not dwell deep into the leaves. It has many categories like: organelle, cell, cellpart, etc. but for example does not have very fine tuned categories like apoptotic signaling pathway is missing.

PFP: Better and more specific than GOTCHA. It had probability values for all the go terms it predicted, the cut-off point was ~20%. In total it made 95 molecular predictions, 260 biological, and 85 cellular.

FFPred: One of the most detailed results, also very visual. Shows High to Low Reliability intervals and probability values. Very specific categories.

Argot: The most detailed one. There are blast hits and HMM models to parse through. There's a score value and internal confidence values etc to tell about the quality of GO's. They are very specific and there's also graphical views of them. Argot was the second slowest to run, but I liked it the most and would use it in future.

Batch annotation with ARGOT2:

```
module load blast+
blastp -outfmt "6 qseqid sseqid evalue " -query only5.fa -db uniprot -out
blast_only5_output -num_threads 10
```

Output example line:

```
W5S_0001	tr|K4FP15|K4FP15_PECSS	0.0
```

We had 250 sequences. I initially tried to do this with 5 sequences, but it took forever and showed no sign of completing, so I redid it with 2.

```
wget ftp://ftp.sanger.ac.uk/pub/databases/Pfam/releases/Pfam26.0/Pfam-
A.hmm.gz

gunzip Pfam-A.hmm.gz

module loadh hmmer/3.0

hmmpress Pfam-A.hmm

hmmscan --tblout hmm_output_only2 Pfam-A.hmm only2.fasta
```

Results example line:

Bac_DnaA PF00308.13 W5S_0001 - 2e-100 334.8 0.1 3.1e-100 334.2 0.1 1.3 1 0 0 1 1 1 1 Bacterial dnaA protein

Line contains all information, tab separated. target name accession query name accession E-value score bias E-value score bias exp reg clu ov env dom rep inc description of target

Then I downloaded the files to my computer, compressed them to .zip, and submitted them to ARGOT2. Examples from ARGOT results:

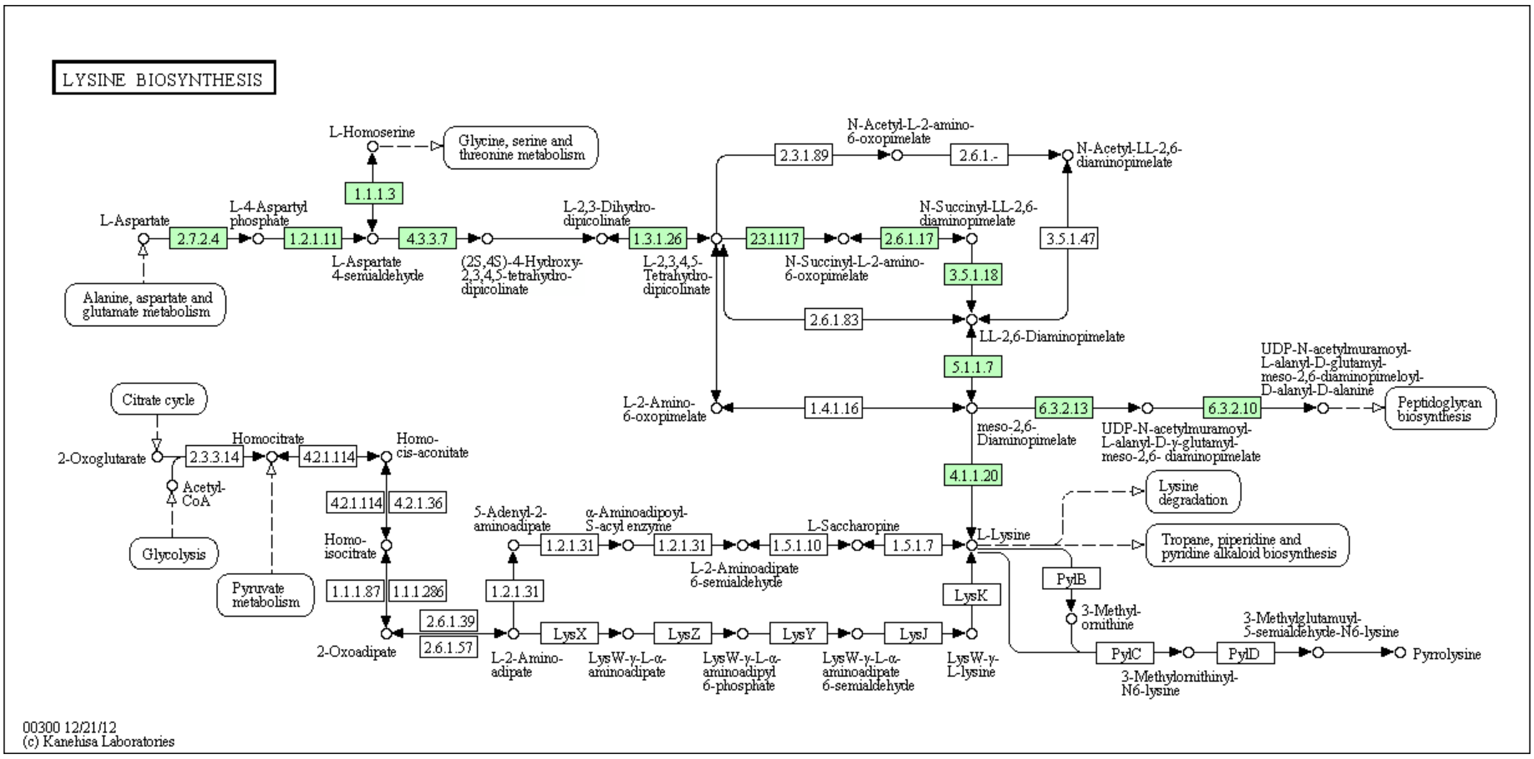
Both proteins had catalytic activity (GO:0003824), and were involved in metabolic processes, and cellular processes. One of the proteins W5S_0002 was part of a macromolecular complex. W5S_0001 was a binding protein.

Pathway mapping:

We had E.Coli sequences and we tried to find out if there's any complete protein pathways that can be built from those sequences. We used KEGG Automatic Annotation Server and mapped the sequences against the KEGG database using BBH.

The Lysine Biosynthesis pathway shows pathways from one metabolism cycle to another. The other pathways weren't as complete as this, but there's also three back to back proteins in Cysteine and Methionine Metabolism, from L-Aspartate to L-homoserine. And in the Citrate Cycle from Citrate to Isocitrate and back.

The best pathway was 00300 Lysine biosynthesis (14):


LYSINE BIOSYNTHESIS
L-Homoserine
Glycine, serine and threonine metabolism
1.1.1.3
L-Aspartate
L-4-Aspartyl phosphate
2.7.2.4
1.2.1.11
4.3.3.7
L-Aspartate 4-semialdehyde
(2S,4S)-4-Hydroxy-2,3,4,5-tetrahydro-dipicolinate
L-2,3-Dihydro-dipicolinate
1.3.1.26
L-2,3,4,5-Tetrahydro-dipicolinate
Alanine, aspartate and glutamate metabolism
N-Acetyl-L-2-amino-6-oxopimelate
2.3.1.89
2.6.1.-
N-Acetyl-LL-2,6-diaminopimelate
3.5.1.47
2.3.1.117
2.6.1.17
N-Succinyl-LL-2,6-diaminopimelate
N-Succinyl-L-2-amino-6-oxopimelate
3.5.1.18
2.6.1.83
LL-2,6-Diaminopimelate
5.1.1.7
UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-meso-2,6-diaminopimeloyl-D-alanyl-D-alanine
Peptidoglycan biosynthesis
L-2-Amino-6-oxopimelate
1.4.1.16
6.3.2.13
6.3.2.10
meso-2,6-Diaminopimelate
UDP-N-acetylmuramoyl-L-alanyl-D-γ-glutamyl-meso-2,6-diaminopimelate
4.1.1.20
Citrate cycle
2-Oxoglutarate
2.3.3.14
Homocitrate
4.2.1.114
Homo-cis-aconitate
Acetyl-CoA
Glycolysis
4.2.1.114
4.2.1.36
Homo-isocitrate
Pyruvate metabolism
1.1.1.87
1.1.1.286
2-Oxoadipate
2.6.1.39
2.6.1.57
5-Adenyl-2-aminoadipate
α-Aminoadipoyl-S-acyl enzyme
1.2.1.31
1.2.1.31
1.2.1.31
L-Saccharopine
1.5.1.10
1.5.1.7
L-2-Aminoadipate 6-semialdehyde
L-Lysine
Lysine degradation
Tropane, piperidine and pyridine alkaloid biosynthesis
PylB
LysK
3-Methyl-ornithine
PylC
PylD
3-Methylglutamyl-5-semialdehyde-N6-lysine
Pyrrolysine
3-Methylornithinyl-N6-lysine
L-2-Amino-adipate
LysX
LysW-γ-L-α-aminoadipate
LysZ
LysW-γ-L-α-aminoadipyl 6-phosphate
LysY
LysW-γ-L-α-aminoadipate 6-semialdehyde
LysJ
LysW-γ-L-lysine
00300 12/21/12
(c) Kanehisa Laboratories

Friday 22.2. RNASeq mapping and analysis

Lecture notes:

FASTQ: 4 riviä per read. @id, seq, + (id), quality-string (ASCII koodi käännettävissä hexa arvoille)

SAM/BAM: Yksi rivi, tab-välitetty + headerit. CIGAR-tag esim: 8M2I4M …
8match,2Insertion,4Match… Match on mikä vain emäs, ei välttämättä a-a.

Filtteröinti:

- Average Quality (e. MAPQ (Phred))
- Enemmän kuin haluttu määrä tuntemattomia nukleotideja (N).
- Duplikaatit
- Low complexity regions, ym. huonot alueet. (DUST-score. Paljastaa suurella arvolla esim. polyA-hännät tai muuten repetiiviset alueet)
- Virheet, kontaminaatiot (esim. keskiarvoinen GC-content lajilla verrataan readejen keskipoikkeamaan)
- Readien alut ja loput voivat olla huonolaatuisia, moni trimmausohjelma leikkaa vain lopun.

Burrows-Wheeler transform. Mappers: BWA, Bowtie

- Suffix tree, suffix array. (Substring from any position from the end of the string)
- Suffix tree: Juuresta lähtee jokainen alisekvenssi. Suuri rakenne, vie paljon muistia.
- Suffix array: Järjestetty array, kaikkien suffixien stringi ei ole haettavissa vaan niiden alkuindeksi.
- B-W: suffixi sylinteri (kaikki permutaatiot käydään läpi), viimeinen kolumni sylinterisanasta on BW, järjestetään aakkosjärjestykseen niin, että sisäinen järjestys säilyy (mitkä kirjaimet ympäröivät). Kaksiulotteinen matriisi, josta
    - Muistiin tallennetaan: BW-arvo ja järjestetty. Eli tunnetaan emäs ja aina sen viereinen kirjain. Koko string voidaan käydä läpi seuraamalla viereistä arvoa päätepisteeseen asti ($).
        - Etsitään GAT. -> Etsitään järjestetystä arraystä kaikki T:t, (etsi T:t joiden järjestämättömässä arrayssä viereinen (suunta!) kirjain on A), …

Tophat, käyttää Bowtietä mappaamiseen. Etsii introni-eksoni-rajoja.

- e. Readi linjautuu kahteen alueeseen referenssissä. (Jakautuu kahteen eksoniin).

Exercizes:

Load all needed programs: samtools, bwa, tophat, cufflinks, and fastx.

Filter reads that have phred quality of under 30 and base quality under 33:

```
fastq_quality_filter -Q 33 -q 30 -i
Cinx_PoolA_RNAseq_GCCAAT_L001_R2_001_FILTERED.fastq -o
Cinx_PoolA_RNAseq_GCCAAT_L001_R2_001_FILTERED.fastq_high_Q

fastq_quality_filter -Q 33 -q 30 -i
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq -o
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq_high_Q
```

Phred quality of 30 is 1/1000 chance of being wrong. It is widely accepted as a good cutoff.

Mapping with BWA:

Bwa indexing, bwa aln generates SA coordinates for the reads, bwa sampe makes paired end alignment for the reads:

```
bwa index Cinxia_scaffold.fasta

bwa aln Cinxia_scaffold.fasta
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq_high_Q >
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq_high_Q.sai

bwa aln Cinxia_scaffold.fasta
Cinx_PoolA_RNAseq_GCCAAT_L001_R2_001_FILTERED.fastq_high_Q >
Cinx_PoolA_RNAseq_GCCAAT_L001_R2_001_FILTERED.fastq_high_Q.sai

bwa sampe Cinxia_scaffold.fasta
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq_high_Q.sai
Cinx_PoolA_RNAseq_GCCAAT_L001_R2_001_FILTERED.fastq_high_Q.sai
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq_high_Q
Cinx_PoolA_RNAseq_GCCAAT_L001_R2_001_FILTERED.fastq_high_Q > aln.sam
```

Using samtools convert to bam:

```
samtools view -bS aln.sam > aln.bam
```

If you want to view the .bam file, you need to further sort and index it to view it in Artemis.

```
samtools sort aln.bam sorted

samtools index sorted.bam
```

If my reads were longer than 200bp, I'd have to use BWA-SW or BWA-MEM. BWA-MEM is the latest and the recommended one.

Mapping with Bowtie:

Alignment and indexing with Bowtie:

```
bowtie2 -x Cinxia_scaffold.index -1
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq_high_Q -2
Cinx_PoolA_RNAseq_GCCAAT_L001_R2_001_FILTERED.fastq_high_Q -S bowtie.sam

bowtie2-build Cinxia_scaffold.fasta Cinxia_scaffold.index
```

Discovering exon junctions with Tophat:

```
tophat Cinxia_scaffold.index
Cinx_PoolA_RNAseq_GCCAAT_L001_R1_001_FILTERED.fastq_high_Q,Cinx_PoolA_RNAseq_GCC
AAT_L001_R2_001_FILTERED.fastq_high_Q
```

To view the result file: go to tophat_out folder and check your result file accepted_hits.bam.

If you want to turn the .bam file to a .sam file, you use samtools:

```
samtools view -h accepted_hits.bam > accepted_hits.sam

(-h keeps the header for further analysis)

For Artemis:

samtools sort accepted_hits.bam sorted_accepted_hits

samtools index sorted_accepted_hits.bam

Using TView:

samtools tview sorted_accepted_hits.bam ../Cinxia_scaffold.fasta
```

Most reads are aligned towards the end of the reference, most of the reference is blank. TopHat suggests splice cites in places where there peaks in coverage of reads.

Cufflinks:

```
cufflinks -o cufflinks_output accepted_hits.bam

../../../../v/users/ahola/bin/maker/bin/cufflinks2gff3
tophat_out/cufflinks_output/transcripts.gtf > transcripts.gff3
```

```
 --overlap-radius          maximum gap size to fill between transfrags (in bp)  [ default:    50 ]
```

If the gap between two units is larger than n, they are combined.

I tried using it with 10 and 100.

```
cufflinks --overlap-radius 100 -o cufflinks_with_overlap100
accepted_hits.bam

../../../../v/users/ahola/bin/maker/bin/cufflinks2gff3
tophat_out/cufflinks_with_overlap100/transcripts.gtf >
transcripts100.gff3

cufflinks --overlap-radius 10 -o cufflinks_with_overlap10
accepted_hits.bam

../../../../../v/users/ahola/bin/maker/bin/cufflinks2gff3
cufflinks_with_overlap10/transcripts.gtf > transcripts10.gff3
```

But I didn't notice any difference in Argo. ? ☹

Adding gene predictors as input to Cufflinks:

```
  -G/--GTF                      quantitate against reference transcript
annotations

  -g/--GTF-guide                use reference transcript annotation to
guide assembly
```

Masking:

```
  -M/--mask-file                ignore all alignment within transcripts in
this file
```

Files Cufflinks generates:

transcripts.gtf : file to be visualized

isoforms.fpkm_tracking : Expression values for the transcripts expressed

genes.fpkm_tracking: Expression values for the genes expressed

skipped.gtf : reads that weren't mapped

Friday 1.3. Comparative Sequence Analysis

Lecture Notes:

Problems:

- sequence sampling (what sequences we should choose? is it worth it to have many very similar sequences (due to evolutionary time) or should there be more variety?)
- what level of conservation is statistically significant?

Solutions:

- codon models: separates mutation process from selection, corrects for relatedness
  - selection happens in aminoacids.
  - Completely different codons rated 0 to empty the matrix and reveal codon clusters (synonymous substitutions -> same aminoacid).
  - omega value for mutations: harmful < 1 (reduces value), neutral = 1, w > 1 (increases value -> big signal). Most mutations are harmful.
    - Omega estimation:
      - Branch: e.g. gene duplication -> new function.
      - Site: Assumes that all sequences have same function

Maximum Likelihood:

Using observation calculate the probability of p. Maximise (L) by finding the value of p that maximizes the function. (occurrences of p)/(all occurrences).

- In phylogenetics: try to find tree that maximizes the probability of the sequence data -> produces a “model” that best describes the data.

Substitutions are stochastic processes:

- mechanistic or empirical models to evaluate evolutionary distance between two sequences.
  - mechanistic model: matrix with values for substitutions (A->T == T->A).
  - Amino acid models:
    - Frequencies estimated from data.
    - f.ex. PAM/BAM
  - For nucleotide sequences:
    - Take into account different things like: transitions/transversions, GC% etc…
    - HKY85 is a good model. Different rates for transitions and transversions.

Method: SLR

- Sitewise
- Estimates common parameters from full data. Assumes all the sites are neutral and tests is that site different than 1. -> Produces confidence intervals and point estimates.
  - Results sites that are under selection (pos/neg).
    - If upper limit is under 1, under negative selection.
    - If upper limit is above 1, under positive selection.

Problems with SLR:

- Produces an alignment with no gaps (usually viral sequences have lots of gaps).
- Alignment is an estimate

Genetic evolutionary distance:

- needs to be corrected for multiple hits (Jukes and Cantor)
- Correction increases when p increases.

Exercizes:

BioLinux:

Reduce memory requirement for BioLinux to orange range, or it will crash when it hogs up too much memory.

Login to Biolinux -> gz/tar file.

Biolinux is basic Ubuntu base. There's a graphical menu for bioinformatics programs that are preinstalled, and also access to normal Ubuntu programs like gedit and Unix command line terminal. All commonly used packages and libraries are preinstalled. You can run it live or install it on your computer.

HIVPol

Used the visualization programs like Jalview and Seaview in Biolinux to visualize alignments and Forester and Treeview to view trees from the HIVpol dataset. The pdf files can be visualized with "evince pdfname".

I tried to understand the plotSlr.R perl-script, but I don't know any R. All I can say is that it saves to a pdf-file. Using the R-script:

```
R --vanilla < plotSlr.R
```

Indelible is a data simulator that generates trees and sequences based on evolutionary models you give it. It can generate nucleotides or proteins, and you can make as many sequences that branch as many times as you want. Execute indelible with command "indelible". LOG.txt describes what values were used as input (ready file made by Ari) and what the values are. All the values are unique (generated random).

- Indelible file is rile ready with preset values created by Ari Löytynoja.
  - [MODEL] 50 bins with 0.02 probability, 50 values of omega
  - [treedepth] 1.4396 (evolutionary time, lower number lower differences, less gaps etc.)
  - [PARTITIONS] 300
  - Output -> Fasta

`bp_translate_seq jordan_small_TRUE.fas > jordan_small_TRUE_pep.fas` to make the nucleotide sequence (codon alignment) to a protein sequence.

OR

```
seaview Jordan_small_TRUE.fas
```

Props -> View as Proteins -> Save

```
Look at the file jordan_small_RATES.txt that lists the rate categories used
across the sequence sites. Category 0 is the most conserved and categories
47--49 are the ones with omega greater than 1. On command line, try
commands
 'grep ^[1-9] jordan_small_RATES.txt |sort -k 2 -n|less' and
 'grep ^[1-9] jordan_small_RATES.txt |sort -k 2 -n -r|less'
```

```
grep ^[1-9] jordan_small_RATES.txt |sort -k 2 -n|less >
jordan_small_RATES_sorted.txt
```

- ➔ sort by column "Class" (Column number 2) (smallest to largest) And pipe it to less for showing.

```
grep ^[1-9] jordan_small_RATES.txt |sort -k 2 -n -r|less
```

- ➔ sort by column "Class" (2) (largest to smallest). And pipe it to less for showing.

SLR data

Convert format with PRANK:

```
prank -convert -d=jordan_small_TRUE.fas -o=jordan_small_TRUE -f=paml
```

Good way of viewing information is with awk:

```
awk 'BEGIN{OFS="\t";print "Site\tOmega\tlower\tupper";} /[0-9]/ &&
$5>1 {print $1,$4,$5,$6}' out.res

Site      Omega     lower     upper

20        99.0000   2.1416    99.0000

85        2.1453    1.2072    3.7392

176       1.9224    1.0442    3.4083

234       1.7717    1.0204    2.9627

433       1.9598    1.0620    3.5245

671       2.6573    1.3577    5.2959

680       2.2348    1.1610    4.4409
```

Here there's a table created with fields Site/Omega/Lower/Upper and the lines where the field Lower is ($5) larger than 1 are chosen and then the correct fields are printed to the table ($1, $4, $5, $6).

Realignment with ClustalW:

```
bp_translate_seq jordan_small.fas > jordan_small.pep

clustalw -infile=jordan_small.pep -usetree=small_tree.tre -
outfile=jordan_small_clustalw_pep.fas -output=fasta
```

Small_tree is the guidetree for ClustalW. Back translation of the alignment, output in paml format.

```
prank -d=jordan_small_clustalw_pep.fas -dna=jordan_small.fas -
o=jordan_small_clustalw_nuc -f=paml
```

Modified slr.ctl to seqfile : Jordan_small_clustalw_nuc.phy and output to clustal_out.res . Then I ran SLR on the new values. Looking at the results:

```
biolinux@biolinux-VBox:~/exercises/jordan$ awk 'BEGIN{OFS="\t";print
"Site\tOmega\tlower\tupper";} /[0-9]/ && $5>1 {print $1,$4,$5,$6}'
clustalw_out.res
Site      Omega     lower     upper
94        3.7129    1.1593    99.0000
389       53.0941   1.3077    99.0000
396       20.6611   1.1737    99.0000
483       40.6983   1.5575    99.0000
Example of a true positive selection not present in the clustal
results.
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^94
jordan_small_RATES.txt
94        18        1
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^389
jordan_small_RATES.txt
389       10        1         INSERTION
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^396
jordan_small_RATES.txt
396       18        1
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^483
jordan_small_RATES.txt
483       12        1         INSERTION
```

Low values in the field that indicates pos. selection. No true positives, all false positives.

Example of a true positive:

```
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^176
jordan_small_RATES.txt
176       47        1
```

This tells to convert positions (--pos) from alignment clustalw_pep.fas (from) to those in alignment outputname_TRUE.prot.fas (to) using species Homo.* as reference (ref).

```
./indexconvert.pl --from jordan_small_clustalw_pep.fas --to
jordan_small_TRUE_pep.fas --ref Homo --pos 94,389,396,483
```

jordan_small_clustalw_pep.fas jordan_small_TRUE_pep.fas:

Alignment with MAFFT:

```
mafft jordan_small.pep > jordan_small_mafft_pep.fas

prank -d=jordan_small_mafft_pep.fas -dna=jordan_small.fas -
o=jordan_small_mafft_nuc -f=paml
```

Modified slr control filet to jordar_small_mafft_pep.fas and mafft_out.ras. Then ran SLR.

When comparing the MAFFT and CLUSTALW Alignments by eye, they are significantly different. CLUSTALW alignment has very little gaps.

```
awk 'BEGIN{OFS="\t";print "Site\tOmega\tlower\tupper";} /[0-9]/ && $5>1 {print
$1,$4,$5,$6}' mafft_out.res
Site        Omega       lower       upper
71          2.0381      1.1640      3.5898
113         4.7205      1.0036      99.0000
145         2.0145      1.0550      3.6663
146         99.0000     1.3772      99.0000
156         99.0000     1.1144      99.0000
349         2.0823      1.1266      3.8280
505         99.0000     1.2735      99.0000
536         2.3020      1.1756      4.5860
541         1.9505      1.0020      3.7580
grep ^71 jordan_small_RATES.txt
71          9           1           INSERTION
grep ^113 jordan_small_RATES.txt
113         19          1           INSERTION
grep ^145 jordan_small_RATES.txt
145         11          1           INSERTION
grep ^146 jordan_small_RATES.txt
146         10          1
grep ^156 jordan_small_RATES.txt
156         30          1           INSERTION
grep ^349 jordan_small_RATES.txt
349         11          1
grep ^505 jordan_small_RATES.txt
505         36          1           INSERTION
grep ^536 jordan_small_RATES.txt
536         22          1
grep ^541 jordan_small_RATES.txt
```

Results are slightly better than CLUSTALW. 156 and 505 are relatively larger values, but neither is enough to count as “positive”.

Alignment with Prank

```
prank13 -d=jordan_small.fas -t=small_tree.tre -translate -F -once -
o=jordan_small_prank_aa_F

prank13 -d=jordan_small.fas -t=small_tree.tre -codon -F -once -
o=jordan_small_prank_codon_F

prank -convert -d=jordan_small_prank_aa_F.best.nuc.fas -
o=jordan_small_prank_aa -f=paml

prank -convert -d=jordan_small_prank_codon_F.best.fas -
o=jordan_small_prank_codon -f=paml
```

The codon alignment is even slower as PRANK now needs to do 61^2 computations for each cell instead of 20^2 computations. Codon alignment can be done with command. Where do the numbers 61^2 and 20^2 come from?

- 61 is the amount of possible codons and 20 is aminoacids.

```
biolinux@biolinux-VBox:~/exercises/jordan$ awk 'BEGIN{OFS="\t";print
"Site\tOmega\tlower\tupper";} /[0-9]/ && $5>1 {print $1,$4,$5,$6}'
prank_codon_out.res
Site      Omega     lower     upper
19        99.0000   1.8011    99.0000
83        1.8810    1.0463    3.2712
176       2.8530    1.4819    5.5349
429       2.0290    1.0791    3.7030
665       2.5657    1.3115    5.1088
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^19
jordan_small_RATES.txt
19        46        1
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^83
jordan_small_RATES.txt
83        14        1         INSERTION
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^176
jordan_small_RATES.txt
176       47        1
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^429
jordan_small_RATES.txt
429       30        1         INSERTION
biolinux@biolinux-VBox:~/exercises/jordan$ grep ^665
jordan_small_RATES.txt
665       17        1         INSERTION
```

There's two really good scores (19, 176) that are sure true posotives, then there's a pretty good score 429. Prank Codon has done the best job so far.

Prank Amino Acids:

```
awk 'BEGIN{OFS="\t";print "Site\tOmega\tlower\tupper";} /[0-9]/ && $5>1
{print $1,$4,$5,$6}' prank_aa_out.res
Site      Omega     lower     upper
19        99.0000   1.6697    99.0000
83        2.1977    1.2605    3.8284
409       2.0855    1.1444    3.7418
627       2.4341    1.2316    4.8794
632       2.0451    1.0542    3.9575
grep ^409 jordan_small_RATES.txt
409       10        1         INSERTION
grep ^627 jordan_small_RATES.txt
627       11        1         INSERTION
grep ^632 jordan_small_RATES.txt
632       19        1         INSERTION
```

Gave two same ones 19 and 83. This was slightly less successful that codon alignment, but still better than clustal and MAFFT.

The results were in line with the results in the paper Ari presented. I expected to have more hits with MAFFT and PRANK.

Ensembl GeneTree:

To retrieve GeneTrees from Ensembl, you can query the database using Python/Perl/etc. We had a ready Perl Script that can do it. It can be executed with ./ensemble_rest_genetree.pl . To retrieve a tree you need to find its GeneTree ID and change that field in the script and rerun it. You can also retrieve a GeneTree with wget and curl:

```
curl
'http://beta.rest.ensembl.org/genetree/id/ENSGT00390000003602?nh_format=s
imple' -H 'Content-type:text/x-nh'

curl 'http://beta.rest.ensembl.org/genetree/id/ENSGT00390000003602?' -H
'Content-type:text/x-phyloxml+xml'
```

wget `http://beta.rest.ensembl.org/genetree/id/ENSGT00390000003602?`

Example: BRCA2 Genetree ID is ENSGT00390000003602.

BRCA1 ID is ENSGT00440000034289.

A GeneTree contains phylogenetical information based on that gene. It has statistics for numer of genes in the tree, speciation noted amounts, duplications, gene split events etc. Human BRCA1 is most closely related to Chimp and then Gorilla.

You can include paralogs or view the gene only.