Week 3: 11th May – Pipelines and workflows

Workflows, how to parallelise jobs

We will use ruffus, python package – a way to manage workflows

Workbenches give you a graphical way to do workflows without programming

DRMAA – interaction with HPC clusters – API that allows software (python) to submit jobs to HPC cluster that is supported by software

Transform merge, split are basic operations

GTF format: genes, chromosome coord, split by chr, count entries by chr and then merge coutns to work out average genes per chr

Write tasks/functions

Def name(input file, output):

Open file then for line in …

For transform we can do any command line statement within ‘’’here’’’

Python decorators

Main statement -runs main script from cgat core

R in front of the string means

@follows #means don’t run this until you’ve run these functions

Pass means don’t do anything

@ these functions are decorators, I think they act as wrappers: They act like stdin/stdout to pass info/data from one function to another?

Day 2: RNAseq pipeline with DS

Move fastq files into new folder (I used a symbolic link here)

/ifs/obds-training/apr20/rose/pipelines/rnaseqpipeline

#Make some tester files with 1000 lines

zcat ERR1755082\_2.fastq.gz | head -n 1000 | gzip > test\_2.fastq.gz

zcat ERR1755082\_2.fastq.gz | head -n 1000 | gzip > test\_2.fastq.gz

run it with python <script.py> make -v 5

Output will give yout html files

Next we could do multiQC to make a nice report of them all

Or we could go ahead to trimmomatic

trimmomatic PE -threads 4 -phred33 -trimlog trimlogreal.txt -summary statssummarytrimreal.txt ERR1755087\_1.fastq.gz ERR1755087\_2.fastq.gz -baseout ERR1755087\_.fastq ILLUMINACLIP:TruSeq3-PE-2.fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36

Day 3

First sorted out all the github problems

I created a repository called obdsRNAseqpipeline on GitHub

Created an empty folder and then clones using this:

git clone git@github.com:rodgson543/obdsRNAseqpipeline.git

#This is my private repository

Next

We are going to go through pseudo alignment today and then do a pipeline

Alignment free method – instead of mapping/QC and counting phase, we do this all in one step: pseudo alignment

Difference is that we specify which transcript the read is from

Good for counting, v fast. matrix is output (can work on this directly)

Transcript per million (so normalised to num of transcript)

More accurate than traditional method

Limitation – cant use it to detect novel transcrpts or sequence variations, ie splicing

Good for DGE

Doesn’t produce BAM file – but can force on CL arguments

Sailfish, salmon, kallisto – Tools, widely used in RNAseq

De-Bruijn lab – transcripts of genome, breaks the transcripts into kmer

Sequences of length k (kmer) default is 31

Each kmer associated with set of transcripts

See pic below

Each path through the graph is a single transcript

Unix recap

Stdin #where unix commands listen for input

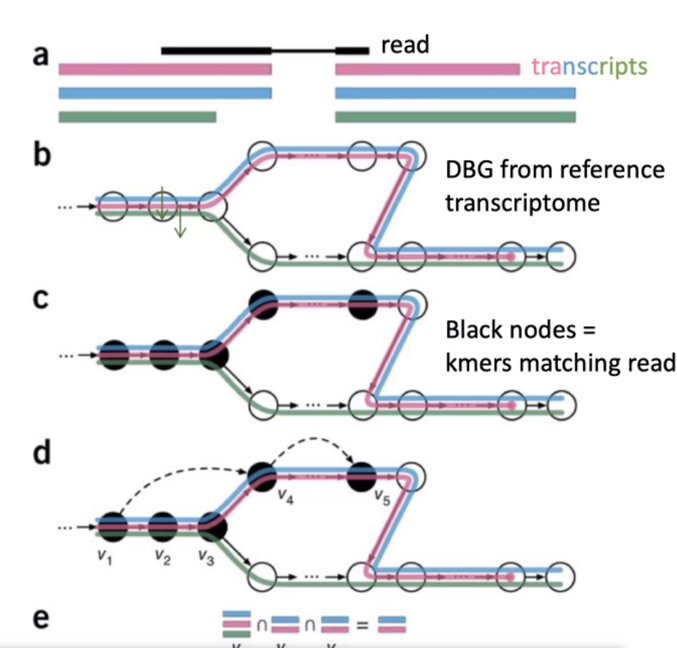
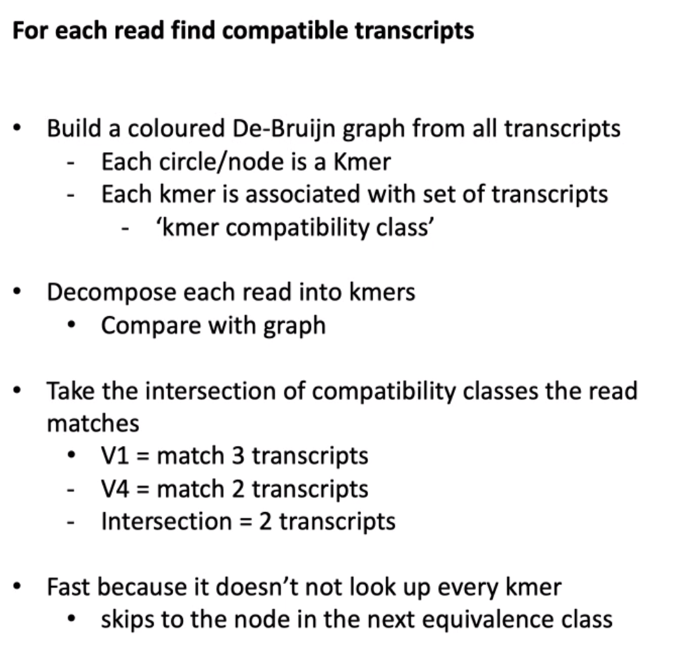
Stdout #unix commands stores output

Stderr #error messages

| #takes stdout from one to stdin to another

>#takes stdout into a command/file

<



Kmers (black circle) – are they equivalent across sequence or exon etc – ie. Green transcript is missing some sequence (ie an exon), then join up at end. Pink transcript terminates earlier (missing an exon)

For each read – in c – black nodes are kmers

Each node is a compatibility class, take intersection – which transcript the read comes from (ie only blue or pink, not green)

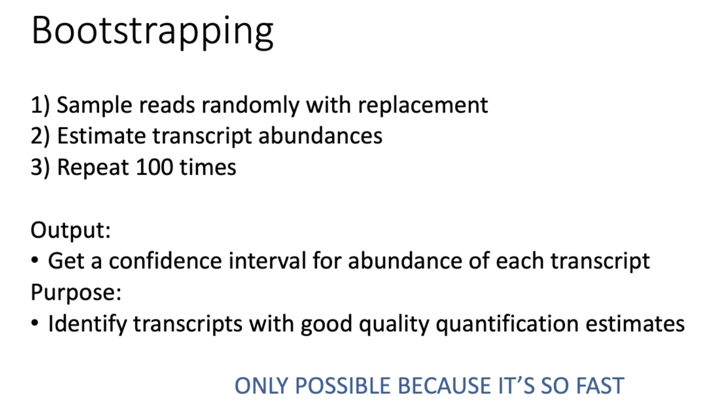
Ie. When you index a genome for mapping, before you start mapping reads for pseudoalignment, build a ref genome

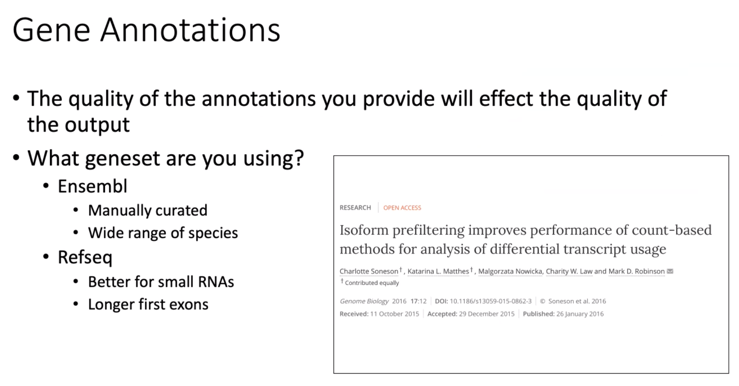
Logical splitting of the sequence into sub sequences of certain length – allows you to only really look at those kmers where they are different.

The reference will be either a ref genome or can build your own

Bootstrapping – randomly sample a subset of reads

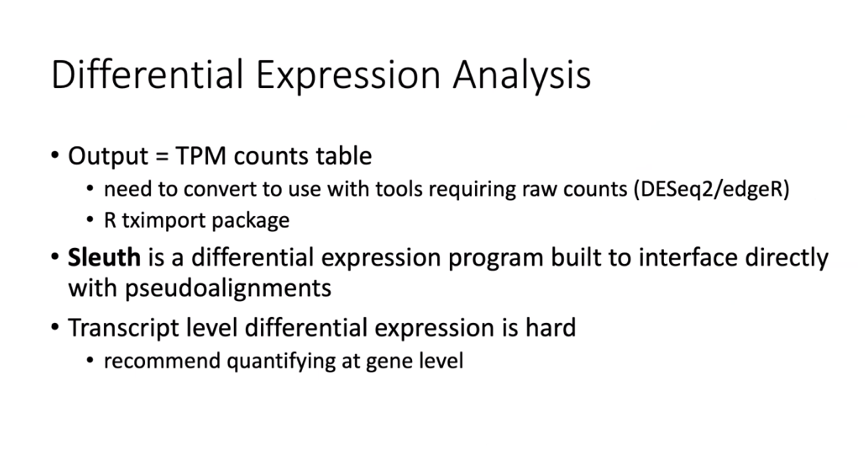
Perform 100 times, estimate transcript abundance each time (gives a confidence interval- ie erro) advantage over mapping method.



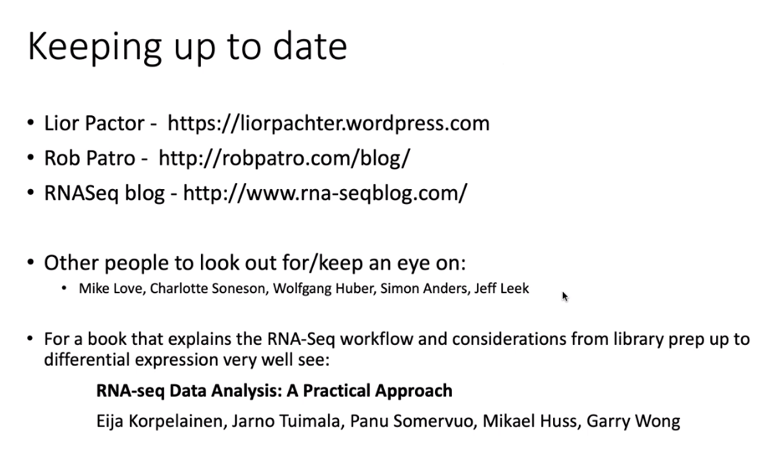


DGE analysis – from pseudoalignment (transcripts per million) have to convert TPM back to counts. Or can use sleuth (directly from kallisto – takes in tpm and confidence intervals)

Best to work at gene level



Leading the field:



Today we’re going to implement this pipeline:

How to start: look at slide 2:

<https://tinyheero.github.io/2015/09/02/pseudoalignments-kallisto.html>



Week 3 – Thursday – CHIP-seq

Today we’re going to go through background of CHIPseq

Then we will read through someone elses pipeline and see if we can understand it

Background:

Crosslink proteins to dna, lyse cells, pulldown, enrich for protein of interest, dna then sequence (illumine) ie. Transcription factors or peaks – ie chromatin marks that will give you info on functional state

Andrea has a SNP within a promoter region, think it influences TF binding – but she doesn’t know the TF. Jasper (based on motif in seq) – detect anything that binds the motif.

Can provide 1-2000 bp for promoter region

Bias: CHIP can be tricky on small/inconstant numbers of cells.

Library fragmentation – (sonication -biases open chrom)

Tagmentation (inset seq into genome into genome – sequence bias)

Nucleosomes -mnase

Antibody that you use has to be specific – use several antibodies for epitope covering etc

Controls

Antibody, no chrom and vv – but what’s the point in sequencing

Input control – chromatin not immunoprecipitated – fragmentation bias is covered here. So you can get same overrepresented places where fragmentation has happened so more reads. But you need high sequencing as you’re covering a lot of the genome

Lost internet.

Normalise across conditions (spike in) – if everything’s same pattern but lower in one condion than another – specific to experimental design

Need to just identifiy where in the genome the reads are coming from. They use PE as better for duplicate removal – shorter read length

Greater for input. – depth depends on your experiment

Encode symposium – give guidelines based on their data

Workflow

fastq – fastqc

mapping – going to use bowtie2 – doesn’t take splicing into account

mapping qc – samtools/picard

peak calling – including macs2

QC of peak calling – Chipseeker is R so probably will use something else

Motif analysis – Homer/beam

Peak annotation – chip seeker-tels you where nearest gene is, whether peaks fall into gene/enhancer more than you’d expect

DEseq2 – DGE for differential binding

MAPPing qc

Sequencing of same fragment multiple times – might see a peak where there isn’t one – remove duplicates

Q – mitochondrial duplicates

Reads that are not properly paired – doesn’t make sense basically

Peak calling is basically seeing the pileup of reads – more than control in certain regions (and more than surrounding region)

Pooling samples – increase calling ability – look at overall properties of dataset.

Downsampling can help with this – but can be complicated

Impossible to normalised based on number of reads? Can try different ways –

MACS2 – industry standard – we will use this one today.

Check reproducibility – ie bedtools, overlap peaks between replicates

Encode project – irrep discovery rate – correlation of ranks between replicates – where does this fall away? Throw away weaker peaks. Controversional – only works with 2 replicates. David Sims doesn’t like it

ChipSeq – black listed – things that just always come up

Peak annotation – homer/great

Nearest neighbour. Promoters are easy to – but enhancers can be a long way from target genes so who knows. Caveeats of this – ie closest gene to chipseq peak may not be correct

Counting is same as DGE

How many reads overlap peaks

Want 3 replicates at least – diffbind – provides a wrapper – but limited with design of DGE comparison

Motif analysis – online interfaces, running on CL

Meme suite and homer good for this. – take sequences under peaks (ie under summit of peak) program will look for repeated kmer and will try and find TF motifs – tells you where within the peak the motifs are – association and cobinding of different TF factors

Viewing peaks – should just look at data first, just initially look at bam peaks, memory intensive etc. Convert bam to bed coverage/bigwig – breaks itno region and gives a coverage summary

Bigwig – compressed/indexed – less memory intensive

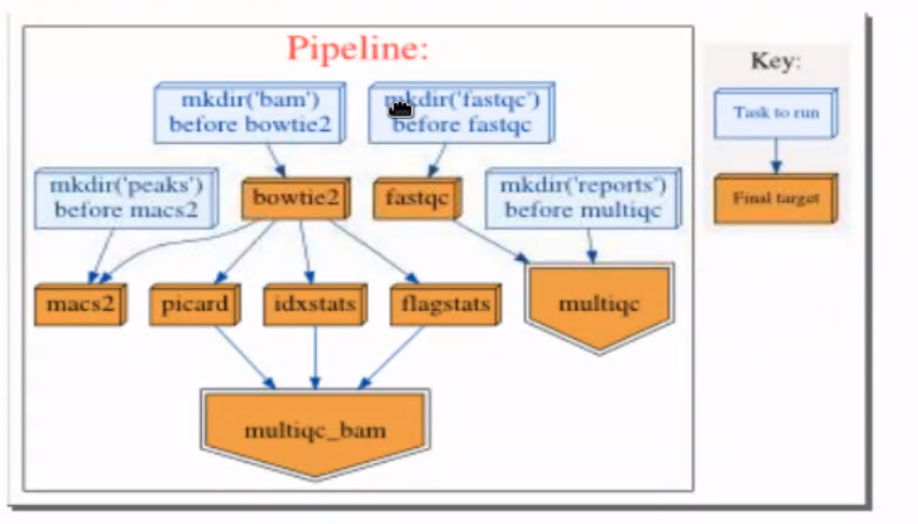
Exercise: we are going to now write the pipeline. Dataset is internal. All have an input control. We have them on the sever

Goes as far as peak calling – git pull the file off Github – chipseq\_pipeline

I’ve started to annotate this now

Shared week 3 – data

We’re going to look at the plot first.

Fastqc/multiqc

Mapping then QC – picard. Idx and flag stats. Draws thm all into multiqc

Then goes to macs2

Add remove duplicate step

Add Filtering – get rid of bad reads

Add merge step – merge controls and treatments

Fell over on macs2 – our py3 conda hasn’t been added into our environemt so I did conda install macs2

Also conda install homer

#Adding a deduplication before peakcalling – before macs2

Running just the multiqc\_bam   
python script multiqc\_bam -v 5

Picard

Samtools idxstats chr x or y tells you whether samples are male or female.

Duplicates – we need to get rid of some of it

Duplicates are reads that map in same coordinates – both pairs map to same cooridinates as another pair. PCR – amplification bias. Throw em away as assume they’re coming from same piece of genomic DNA. UMI are only real way to see if they come from an independent DNA molecule

All of mapping QC gets done at same time – but would be nice to add this info into the multiQC report.

MarkDuplicates -

Filter stage

Post duplicate removal – filter the bam file. – get rid of unpaired things, secondary alignment, mitochondrial reads

<http://broadinstitute.github.io/picard/explain-flags.html>

sam flags page

<https://davetang.org/wiki/tiki-index.php?page=SAMTools>

Had to do the bam indexing manually here – the pipeline didn’t work for some reason.

Currently the macs2 has been done on the bowtie2 data,

Initially after bowtie that we also do the remove duplicate, and filtering on this then

So we’ve done the macs (peak calling) on the output of the bowtie2- which is the alignment (without removing the duplicates or doing any of the filtering)

* Should we also do the broad peaks?
* Filter blacklisted peaks (Bedtools)
* •Count reads under peaks (Bedtools)
* •Compare replicate overlap (Bedtools)
* •Annotate peaks (Homer)
* •Motif analysis (Homer)
* •View outputs (IGV)

Ok we first need to filter the blacklisted peaks

Blacklist regions – gives you a whitelisted list – can do before or after – better to do before as you don’t want the blacklisted regions to affect how the model looks in the end

ls -lh \*filter\_read\* #shows you these files and memory etc

first going to count peaks under the reads using bedtools

I’m not sure about the homer stuff – we had to do which homer on command line to find where the genome thing is located

Basically need the full path to run homer.

annotatePeaks.pl peaks/rep1\_dex\_peaks.narrowPeak mm10

n case this helps anyone, finally I was able to download the mm10 genome following what Kevin did; it took me a while to understand/find the correct full path (my brain is very bioinformatician...). Here is what I have used: (you have just to replace your username and maybe you have to change the name of your environment (in my case obds-py3 but maybe in yours is obds\_py3).

perl /ifs/obds-training/apr20/<yourusernamehere>/conda/obds\_conda/envs/obds-py3/share/homer-4.11-1/configureHomer.pl -install mm10

at the end, as Kevin showed, I have the mm10 genome in this path:

/ifs/obds-training/apr20/<yourusernamehere>/conda/obds\_conda/envs/obds-py3/share/homer-4.11-1/data/genomes

Day4 – DNAseq

Whole genome sequencing – expensive

Exome – 2% coding genome, saves a lot of costs. Storage space etc.

Genetic cause of a rare disease – monogenic – inherited mutations etc

Looking at tumour samples to see where cancer has come from – genetic history. Chromosomal aberrations etc.

Instead of peak calling:Call variants – GATK genomic analysis tool kit

Interpret peak calling:

Each read maps with possible fewest mismatches

Allow for polymorphisms/SNPs etc.

BLAST BLAt – general purpose. Not fast. Not efficient

GATK – specifialism

Want to have reliable enough tools to pick up SNPs etc

Broad institute recommends burrows wheeler aligner

Gapped alignments – identify short indels,

Soft clipping – ends of reads don’t have to align to genome – good if low quality etc

INPUT

Fastq – sanger quality score – converts to sam

Sam to bam to indexed

Optimised for human mammalian sequencing, parameters can be used on default

Picard tools

Collect hybrid summary metrics – coverage of target region – bait of RNA/DNA regions –

Exome sequencing has a PCR sequencing. WGS doesn’t. So duplicate read removal reuired as amplification bias.

Picard mark duplicates – sets duplicate flags in bam file

Prioritises base pair with best quality score. Or can retain read pair with longest alignment to genome – ie CIGAR string of SAM file

2nd way of doing duplicate read removal

Output of markDuplicates

Variant calling – GATK

Broad – GATK is gold standard

Preprocessing: map to genome. Map duplicates, base quality recalibration step

Feed into haplotype callers – variant caller.

Joins together these callers – joint calling across all samples across experiment

Downstream – variant filtering, annotation

Base quality score recalibration -improve base quality score. Reduce false positive calls. More confidence that it’s a correct base call – we want to be sure the illumine has done a good job of reporting the quality score.

Take mapped data – they look at the correlation between reported fastq quality socre, with things like position of read, the dinucleotide pair (sequencing chemistry can affect) and impirical quality – mismatches, idea of actual error rate of mapping.

Idea is to recalibrate quality score to reflect quality of data. Ie reduced efficiency at end of reads

Empirical mapping score

Big peak (high quality), then after recal – broader peak with bigger spread. Difference between empirccal score (mapping to genome) v reported quality. Should be no diff, but there is – algorithm normalises it

Haplotype caller – like the peak caller

Assembly of reads in region using debruijn (assembles all possible paths – establishes most likely haplotypes,based on idea there are 2 possibles)

DBG – de bruijn most likely

It’s like the dBG yesterday using the kmers and creating a predictive model of what the haplotype sequences are and where the SNPs are etc

Gives you info about the regions that aren’t SNPs – consistent basically

Processing – across cohort and by individual

Variant calling – likely false positives (technical errors)

We need to train the model – large set of high quality known variants

VCF – variant calling format file is output. Headers and entries. Headers give you fields, processing, info etc. ie. Fileformat, where it comes from, ref genome, info, filter and format fields. Entries in the bottom part

<https://samtools.github.io/hts-specs/VCFv4.1.pdf> - vcf filter flags

Entries are tab delimited, variable info – chromosome – positon – id (id of database of DBsnp database of all known snps. Also microsatellite repeats) – ref allele – alternative allele. – quality score of variant – filter (customisable) – annotations of variant – NS (number of samples) DP is depth – format field bits of data ssociated with variant/individual with variant – NA related to

VCF can be 1 or 1000 individuals

Every variant caller will give a VCF file

Key value pair (NS = value)

Common approach is use several approaches

Joint genotyping v joint allele freq –

Variant annotation

Coding – protein of interest, mismatch – info on how to SNP relates back to the effect on the protein – amino acids etc – algoritms to predict effect

VCF filtering

Most common workflows:

Edu interested in understanding copy number variants

vardict and mutect2 is used for Variant callers by Charlotte – somatic v germline. GATK – diagnosis pipeline, looking for genes and known mutations. Not discovery

VCF – CBRG give VCF files, look for particular – genes- look for how real the variants are based on their quality score. Filter based on annotations – linking the genotype to a phenotype that you know.

Trios: paretns/children, ie de novo (child not parents) hets and homs – use the trio structure – phenotype in child not parent – quality score, how good was coverage;

Db database – info on SNPs and association with different phenotype – like cultural noise

Filter based on variety, pathogenicity – impact on predicted protein, change in nucleotide on protein

Genematcher – variant – matching the gene to phenotype.

Going to take some data from Edu and Charlotte (VCF/Fastq files to do these exercises on)

Charlotte – known genes in bm malignancy, tumour only – fastq paired end tumour only

Out of vep - variant annotation – file html file gives a lot of summary info – variant report – 1.1M sNVs – variant class/count. First comparing to reference. But

From no machine -set up?

Module load bio/IGV

Igv.sh

Load data in

Type a gene in –

Colours identify a mismatch

Looks like we’ve called too many variants

Bcftools does not seem to work in our conda environment – trying to fix this

<http://bioconductor.org/packages/release/bioc/vignettes/TVTB/inst/doc/Introduction.html#7_visualising_data>

cat human\_exome.bed | tr [A-Z] [a-z] > human\_exome\_new.bed

cat human\_exome.bed | cut -f1 | sort | uniq #check for contigs in th file. The last bit didn’t work as the chr names are different