Alternative-splicing detection by NGS

walk-through part

Wen-Dar Lin

Bioinformatics core, IPMB

Preface

- This presentation file is to describe key steps and important notes
  - cufflinks pipeline
  - stringtie pipeline
  - rackj pipeline

Files

- https://maccu.project.sinica.edu.tw/20211007/
  - AS_20211007ws.pptx
    - This file
  - walkthrough_cufflinks_20210922.txt
    - Steps of cufflinks pipeline
  - walkthrough_stringtie_20210922.txt
    - Steps of stringtie pipeline
  - walkthrough_rackj_20210922.txt
    - Steps of rackj pipeline & motif discovery
  - ExampleData.tar.gz
    - Example dataset

The example dataset

- Randomly (1%) extracted from publicly available SRA dataset (SRP071829)
  - created by Dr. Matzke's lab, IPMB
  - triplicates of control and treatment Arabidopsis samples
    - a total of 6 samples
  - strand specific pair-ended RNAseq
    - a total of 12 FASTQ files

Disclaimer

- This part is intended to give useful logs to users who has experiences on using linux environment for computation.
- For those who are not familiar with linux operation, you may take a look at explanations of key steps.
- CAUTION: Better not copy command from this PowerPoint file. Office might twist symbols like - ' ".

Walkthroughs

- A fresh new Ubuntu18 environment was adopted for installing all necessary programs and running all programs
  - System related steps might be different for your environment
  - For a linux system that has been running various programs for a while, some installations might had already been done.
- All steps are with short comments

Cufflinks pipeline

- 0. install Tophat2, Bowtie2, and cufflinks
  - They are all needed for this pipeline

```
### 0. install Tophat2, Bowtie2, and cufflinks

ubuntu@ubuntu18:~$ sudo apt install unzip
ubuntu@ubuntu18:~$ sudo apt install python

(download & install bowtie2)
ubuntu@ubuntu18:~$ wget https://downloads.sourceforge.net/pro

ubuntu@ubuntu18:~$ unzip bowtie2-2.4.4-linux-x86_64.zip

ubuntu@ubuntu18:~$ export PATH=/home/ubuntu/bowtie2-2.4.4-lin
(...)
```

Cufflinks pipeline

- 1. Download data files and preparation

```
### 1. Download data files and prepration

(get exmample data files & extract)
ubuntu@ubuntu18:~$ wget https://maccu.project.sinica.edu.tw/2

ubuntu@ubuntu18:~$ tar -zxvf ExampleData.tar.gz
ubuntu@ubuntu18:~$ cd ExampleData/

(we will align reads using tophat2 so remove existing BAM fil
ubuntu@ubuntu18:~/ExampleData$ rm *.bam
```

Cufflinks pipeline

- 2. running tophat2
  - In addition of building genome index, it is strongly suggest to build transcriptome index by Tophat2 manual.
    - Avoiding race condition and saving time.

```
### 2. running tophat2

(bowtie2 build genome index, this takes time)
ubuntu@ubuntu18:~/ExampleData$ bowtie2-build
TAIR10_chr_all.fas tair10.genome

(tophat2 build transcriptome index, this takes time)
ubuntu@ubuntu18:~/ExampleData$ tophat2 -G
TAIR10_GFF3_genes_transposons.gff --transcriptome-
index=tair10.transcriptome/known tair10.genome
```

Cufflinks pipeline

- 2. running tophat2
  - “ls src/*.fq.gz | sort” will output filenames of FASTQ files (gzipped) to “perl …”
  - The perl oneliner is to pair those paired FASTQ files and produce one tophat2 command for mapping them.
  - In so doing, you don’t have to enter 6 commands for 6 samples. You may simply save the oneliner into your work log without managing a number of scripts.

```
(align reads using tophat2, guided with tair10 annotation)
ubuntu@ubuntu18:~/ExampleData$ ls src/*.fq.gz | sort | perl
-ne 'chomp; /.+\/(.+)_R\d\./; push @{$hash{$1}},$_; if(eof){
for $k (sort keys %hash){ $cmd="tophat2 -o $k"."_tophat2 -p
4 --transcriptome-index=tair10.transcriptome/known
tair10.genome @{$hash{$k}}"; print "\nCMD: $cmd\n"; system
$cmd } }'
```

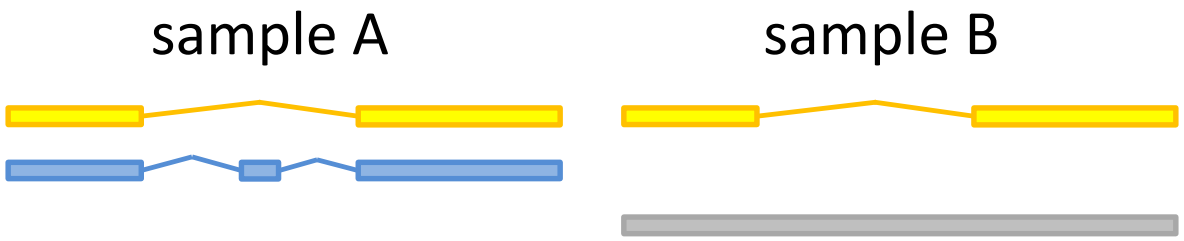
Cufflinks pipeline

- 2. running tophat2
  - When you are using a computing cluster or job scheduler, you may replace the “system $cmd” by some job submission command.
  - Just a personal habit.
  - Remove “system $cmd” to see outputted commands.

```
(align reads using tophat2, guided with tair10 annotation)
ubuntu@ubuntu18:~/ExampleData$ ls src/*.fq.gz | sort | perl
-ne 'chomp; /.+\/(.+)_R\d\./; push @{$hash{$1}},$_; if(eof){
for $k (sort keys %hash){ $cmd="tophat2 -o $k"."_tophat2 -p
4 --transcriptome-index=tair10.transcriptome/known
tair10.genome @{$hash{$k}}"; print "\nCMD: $cmd\n"; system
$cmd } }'
```

Cufflinks pipeline

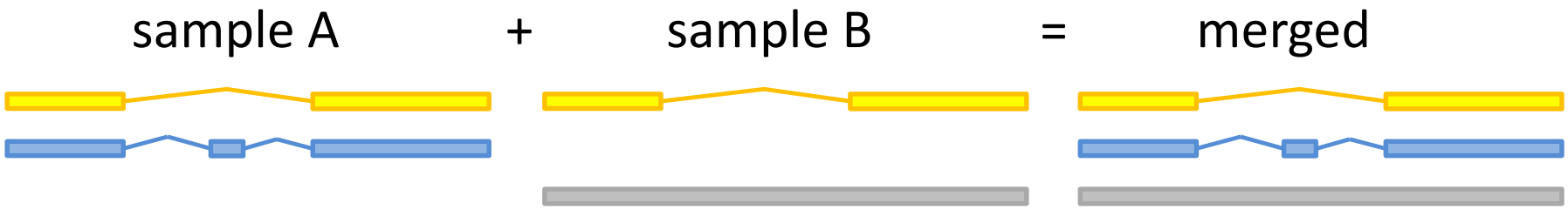
- 3. running cufflinks programs
  - In this very first step, what we have to do is to use cufflinks to build one assembly for each sample




```
(cufflinks, guided assembly)
ubuntu@ubuntu18:~/ExampleData$ ls
*_tophat2/accepted_hits.bam | perl -ne 'chomp;
/(.+?)_tophat2/; $cmd="cufflinks -o $1_cufflinks -p 4 -g
TAIR10_GFF3_genes_transposons.gff $_"; print "\nCMD:
$cmd\n"; system $cmd'
```

Cufflinks pipeline

- 3. running cufflinks programs
  - Comparison between samples means that we need a unified transcriptome. So use cuffmerge to combine assemblies.




```
ubuntu@ubuntu18:~/ExampleData$ cat cuffmerge_gtf.list
control_rep1_cufflinks/transcripts.gtf
control_rep2_cufflinks/transcripts.gtf
(...)
ubuntu@ubuntu18:~/ExampleData$ cuffmerge -p 4 -o cuffmerge -
g TAIR10_GFF3_genes_transposons.gff cuffmerge_gtf.list
```

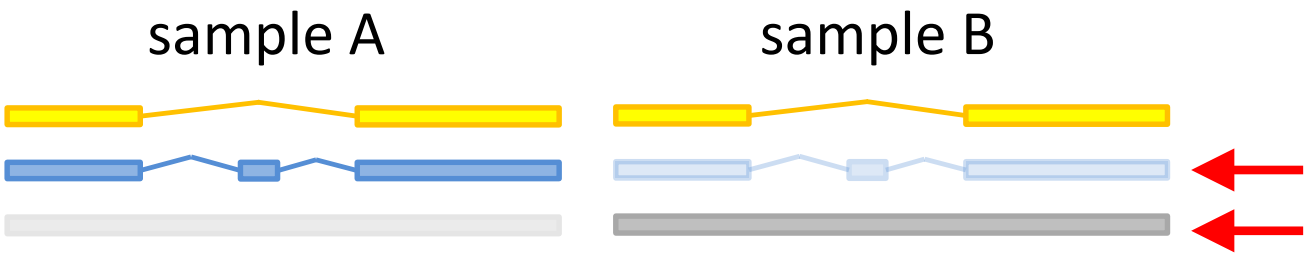
Cufflinks pipeline

- 3. running cufflinks programs
  - Above steps are for assembly generation. With merged assembly, use cuffquant to quantify transcripts for each sample.


```
(cuffquant, guided alignments)
ubuntu@ubuntu18:~/ExampleData$ ls
*_tophat2/accepted_hits.bam | perl -ne 'chomp;
/(.+?)_tophat2/; $cmd="cuffquant -o $1_cuffquant -p 4
cuffmerge/merged.gtf $_"; print "\nCMD: $cmd\n"; system
$cmd'
```

Cufflinks pipeline

- 3. running cufflinks programs
  - With transcripts quantified separately for each sample. Use cuffdiff to predict differentially expressed isoforms.




```
(cuffdiff, compute difference)
ubuntu@ubuntu18:~/ExampleData$ cuffdiff -p 4 -o cuffdiff
cuffmerge/merged.gtf
control_rep1_cuffquant/abundances.cxb,control_rep2_cuffquant
/abundances.cxb,control_rep4_cuffquant/abundances.cxb
treatment_rep5_cuffquant/abundances.cxb,treatment_rep7_cuffq
uant/abundances.cxb,treatment_rep9_cuffquant/abundances.cxb
```

StringTie pipeline

- 0. install Tophat2, Bowtie2, and StringTie
  - They are all needed for this pipeline

```
### 0. install Tophat2, Bowtie2, and cufflinks

ubuntu@ubuntu18:~$ sudo apt install unzip
ubuntu@ubuntu18:~$ sudo apt install python

(download & install bowtie2)
ubuntu@ubuntu18:~$ wget https://downloads.sourceforge.net/pro

ubuntu@ubuntu18:~$ unzip bowtie2-2.4.4-linux-x86_64.zip

ubuntu@ubuntu18:~$ export PATH=/home/ubuntu/bowtie2-2.4.4-lin
(...)
```

StringTie pipeline

- 1. Download data files and preparation

```
### 0. install ### 1. Download data files and prepration

(get exmample data files & extract)
ubuntu@ubuntu18:~$ wget https://maccu.project.sinica.edu.tw/2

ubuntu@ubuntu18:~$ tar -zxvf ExampleData.tar.gz
ubuntu@ubuntu18:~$ cd ExampleData/

(we will align reads using tophat2 so remove existing BAM fil
ubuntu@ubuntu18:~/ExampleData$ rm *.bam
```

StringTie pipeline

- 2. running tophat2
  - In addition of building genome index, it is strongly suggest to build transcriptome index by Tophat2 manual.
    - Avoiding race condition and saving time.

```
### 2. running tophat2

(bowtie2 build genome index, this takes time)
ubuntu@ubuntu18:~/ExampleData$ bowtie2-build
TAIR10_chr_all.fas tair10.genome

(tophat2 build transcriptome index, this takes time)
ubuntu@ubuntu18:~/ExampleData$ tophat2 -G
TAIR10_GFF3_genes_transposons.gff --transcriptome-
index=tair10.transcriptome/known tair10.genome
```

StringTie pipeline

- 2. running tophat2
  - The same “ls src/*.fq.gz | sort” + perl oneliner for executing tophats
  - NOTE: you may switch to any other mapping tools as you like. MUST refer StringTie official website!

```
(align reads using tophat2, guided with tair10 annotation)
ubuntu@ubuntu18:~/ExampleData$ ls src/*.fq.gz | sort | perl
-ne 'chomp; /.+\/(.+)_R\d\./; push @{$hash{$1}},$_; if(eof){
for $k (sort keys %hash){ $cmd="tophat2 -o $k"."_tophat2 -p
4 --transcriptome-index=tair10.transcriptome/known
tair10.genome @{$hash{$k}}"; print "\nCMD: $cmd\n"; system
$cmd } }'
```

StringTie pipeline

- 3. running stringtie programs
  - It seems that stringtie doesn't recognize TAIR10 GFF3 file well. So I decide to translate this GFF3 file into a GTF file.
  - Only exon and pseudogenic_exon records were handled.

```
(translate TAIR10 GFF3 into GTF)
ubuntu@ubuntu18:~/ExampleData$ cat
TAIR10_GFF3_genes_transposons.gff | perl -ne
'@t=split(/\t/); print if ($t[2] eq "exon") || ($t[2] eq
"pseudogenic_exon")' | perl -ne 'chomp; @t=split(/\t/);
$t[8]=~/Parent=(.+?)\.(.+)/; $t[2]="exon"; $t[8]="gene_id
\"$1\"; transcript_id \"$1.$2\";"; print join("\t",@t)."\n"'
> tair10.gtf
```

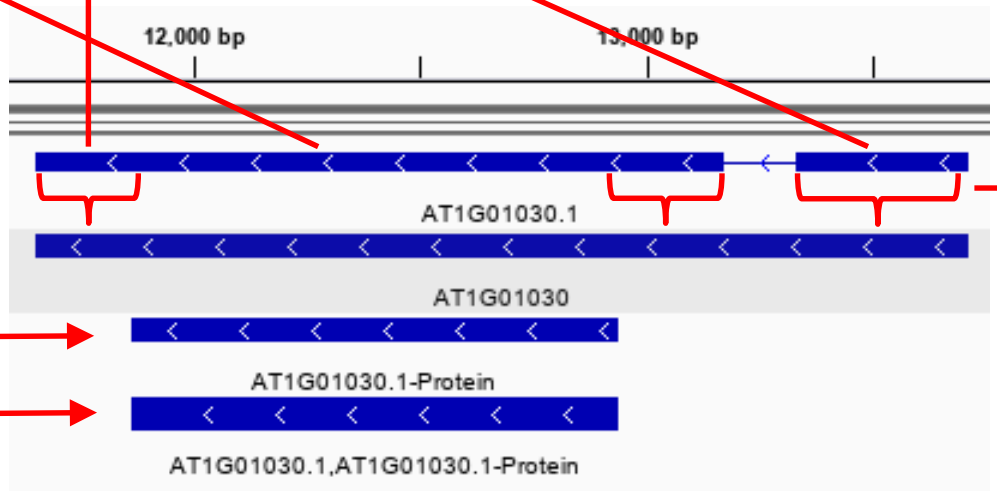
The GFF3 format

- It is common to see that genome annotations are stored in a GFF3 format file
  - Usually in download area of the genome's official website, which should be with the genome's FASTA file.
    - NOTE: if there is a README file, must check it.
- A GFF3 file storing genome annotation tells you *which genes are at where of the genome*.

The GFF3 format

- A GFF3 file also stores hierarchy of recorded objects.

Chr1	gene	11649	13714	ID=AT1G01030
Chr1	mRNA	11649	13714	ID=AT1G01030.1;Parent=AT1G01030
Chr1	protein	11864	12940	ID=AT1G01030.1-Protein;Derives_from=AT1G01030.1
Chr1	five_prime_UTR	13335	13714	Parent=AT1G01030.1
Chr1	exon	13335	13714	Parent=AT1G01030.1
Chr1	five_prime_UTR	12941	13173	Parent=AT1G01030.1
Chr1	CDS	11864	12940	Parent=AT1G01030.1-Protein;
Chr1	three_prime_UTR	11649	11863	Parent=AT1G01030.1
Chr1	exon	11649	13173	Parent=AT1G01030.1

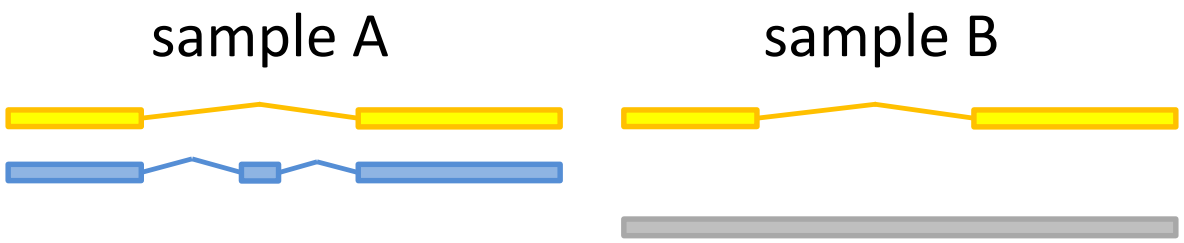


The GTF format

- Can be considered as a previous version of GFF3
  - which usually stores only exon regions
  - plus gene id and transcript id for each exon.

StringTie pipeline

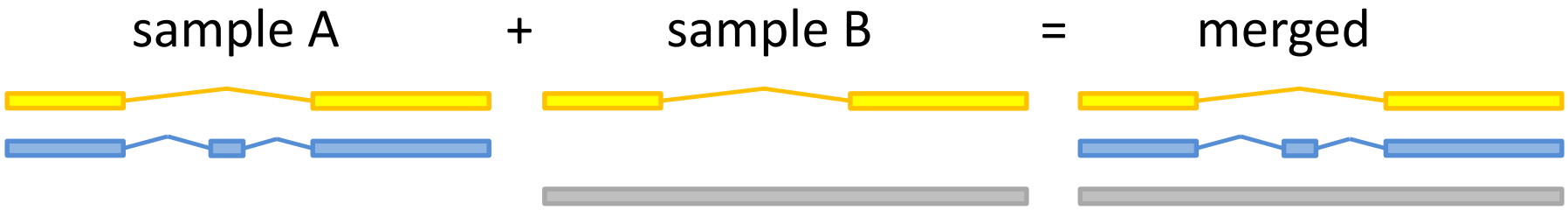
- 3. running stringtie programs
  - In this very first step, what we have to do is to use stringtie to build one assembly for each sample




```
(stringtie, guided assembly)
ubuntu@ubuntu18:~/ExampleData$ ls
*_tophat2/accepted_hits.bam | perl -ne 'chomp;
/(.+?)_tophat2/; $cmd="stringtie $_ -o $1.gtf -p 4 -G
tair10.gtf"; print "\nCMD: $cmd\n"; system $cmd'
```

StringTie pipeline

- 3. running stringtie programs
  - Comparison between samples means that we need a unified transcriptome. So use “stringtie --merge” to combine assemblies.




```
(stringtie, merge mode, combine assemblies of replicates
into one master transcriptome)
ubuntu@ubuntu18:~/ExampleData$ stringtie --merge -G
tair10.gtf -o merged.gtf control_rep1.gtf control_rep2.gtf
control_rep4.gtf treatment_rep5.gtf treatment_rep7.gtf
treatment_rep9.gtf
```

StringTie pipeline

- 3. running stringtie programs
  - Above steps are for assembly generation. With merged assembly, use stringtie with option “-eB” to produce counts for transcripts.
  - NOTE: -o must be assigned to a separate subfolder for each sample because stringtie are outputting all count files with exactly the same names.


```
(stringtie, generate tables)
ubuntu@ubuntu18:~/ExampleData$ ls
*_tophat2/accepted_hits.bam | perl -ne 'chomp;
/(.+?)_tophat2/; $cmd="stringtie $_ -eB -o $1/$1.gtf -p 4 -G
merged.gtf"; print "\nCMD: $cmd\n"; system $cmd'
```

StringTie pipeline

- Count files in the same names

```
ubuntu@ubuntu18:~/ExampleData$ ls -l */*.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2826155 Oct  5 17:11 control_rep1/e2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu 11682721 Oct  5 17:11 control_rep1/e_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2246980 Oct  5 17:11 control_rep1/i2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu  5092894 Oct  5 17:11 control_rep1/i_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  3554980 Oct  5 17:11 control_rep1/t_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2826155 Oct  5 17:11 control_rep2/e2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu 11684916 Oct  5 17:11 control_rep2/e_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2246980 Oct  5 17:11 control_rep2/i2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu  5093349 Oct  5 17:11 control_rep2/i_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  3555067 Oct  5 17:11 control_rep2/t_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2826155 Oct  5 17:11 control_rep4/e2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu 11679732 Oct  5 17:11 control_rep4/e_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2246980 Oct  5 17:11 control_rep4/i2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu  5092408 Oct  5 17:11 control_rep4/i_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  3554974 Oct  5 17:11 control_rep4/t_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2826155 Oct  5 17:11 treatment_rep5/e2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu 11682131 Oct  5 17:11 treatment_rep5/e_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  2246980 Oct  5 17:11 treatment_rep5/i2t.ctab
-rw-rw-r-- 1 ubuntu ubuntu  5093373 Oct  5 17:11 treatment_rep5/i_data.ctab
-rw-rw-r-- 1 ubuntu ubuntu  3554953 Oct  5 17:11 treatment_rep5/t_data.ctab
```

StringTie pipeline

- 3. running stringtie programs
  - The logic of stringtie is to collect counts in all those count files to some other program for differential analysis.
  - We use “prepDE.py” (comes with stringtie package) to generate a table of isoform read counts and sent it to DESeq2 for differential analysis.

```
(generate gene read counts/transcript read counts)
ubuntu@ubuntu18:~/ExampleData$ prepDE.py

ubuntu@ubuntu18:~/ExampleData$ ls *.csv
gene_count_matrix.csv  transcript_count_matrix.csv
```

StringTie pipeline

- 4. R for differential expression
  - To run DESeq2, we have to install R
    - All steps are from R official website for ubuntu

```
(install R for ubuntu, steps from R offical website)
ubuntu@ubuntu18:~$ sudo apt update -qq
ubuntu@ubuntu18:~$ sudo apt install --no-install-recommends
software-properties-common dirmngr
ubuntu@ubuntu18:~$ wget -qO- https://cloud.r-
project.org/bin/linux/ubuntu/marutter_pubkey.asc | sudo tee
-a /etc/apt/trusted.gpg.d/cran_ubuntu_key.asc
ubuntu@ubuntu18:~$ sudo add-apt-repository "deb
https://cloud.r-project.org/bin/linux/ubuntu $(lsb_release -
cs)-cran40/"
```

StringTie pipeline

- 4. R for differential expression
  - To make DESeq2 be installed in R, there are a number of system packages need to be installed
  - NOTE: different packages might be needed under different environment. You have to check error messageS when executing R commands in the next slide
    - No worry, somebody might have had got the same error message. So just google the error message with “DESeq2”

```
(install necessary packages for DESeq2)
ubuntu@ubuntu18:~$ sudo apt install libxml2-dev libcurl4-
openssl-dev libssl-dev libpng-dev libblas-dev liblapack-dev
libgfortran3 gfortran
```

StringTie pipeline

- 4. R for differential expression
  - Enter R, install Bioconductor, and install DESeq2
  - load the library by “library(DESeq2)”

```
ubuntu@ubuntu18:~/ExampleData$ R

(install bioconductor & DESeq2)
> if (!requireNamespace("BiocManager", quietly = TRUE))
+     install.packages("BiocManager")
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
Warning in install.packages("BiocManager") :
  'lib = "/usr/local/lib/R/site-library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library
‘~/R/x86_64-pc-linux-gnu-library/4.1’
to install packages into? (yes/No/cancel) yes

> BiocManager::install("DESeq2")

> library(DESeq2)
```

StringTie pipeline

- 4. R for differential expression
  - Load transcript count CSV file
  - Visual confirm column headers
    - controlx3 and treatmentx3

```
> countData <- as.matrix(read.csv("transcript_count_matrix.csv",
row.names="transcript_id"))
> head(countData)
            control_rep1 control_rep2 control_rep4 treatment_rep5
AT4G04480.1            0            0            0              0
AT1G07730.2            0            0            0              0
AT1G38430.1            0            0            0              0
AT1G03340.1            7            4            4              4
AT2G25040.1            0            0            0              0
AT1G04440.1           52           68           88            106
            treatment_rep7 treatment_rep9
AT4G04480.1              0              0
AT1G07730.2              0              2
AT1G38430.1              0              0
AT1G03340.1              4             11
AT2G25040.1              0              0
AT1G04440.1             85            113
```

StringTie pipeline

- 4. R for differential isoform expression
  - The following R commands should output a CVS file named desqOut.csv, which can be opened directly by Excel
  - CAUTION: the use of condition= c("A","A","A","B","B","B") is only working for comparisons of two conditions

```
> condition= c("A","A","A","B","B","B")
> df = data.frame(condition,row.names=colnames(countData))
> dds <- DESeqDataSetFromMatrix(countData,colData=df,design=~condition)
> dds <- DESeq(dds)
> res <- results(dds)
> write.csv(as.data.frame(res),file="desqOut.csv")
> quit()
Save workspace image? [y/n/c]: n

ubuntu@ubuntu18:~/ExampleData$ head -3 desqOut.csv
"","baseMean","log2FoldChange","lfcSE","stat","pvalue","padj"
"AT4G04480.1",0,NA,NA,NA,NA,NA
"AT1G07730.2",0.322550149365598,1.8429268299077,4.03846570623945,0.45634
```

rackj pipeline

- Here are general description of rackj pipelines
- Steps 0&1: make the environment and data ready
- Step 2: mapping reads to the genome (optional)
- Step 3: a LOT of commands that compute and compare numbers

rackj pipeline

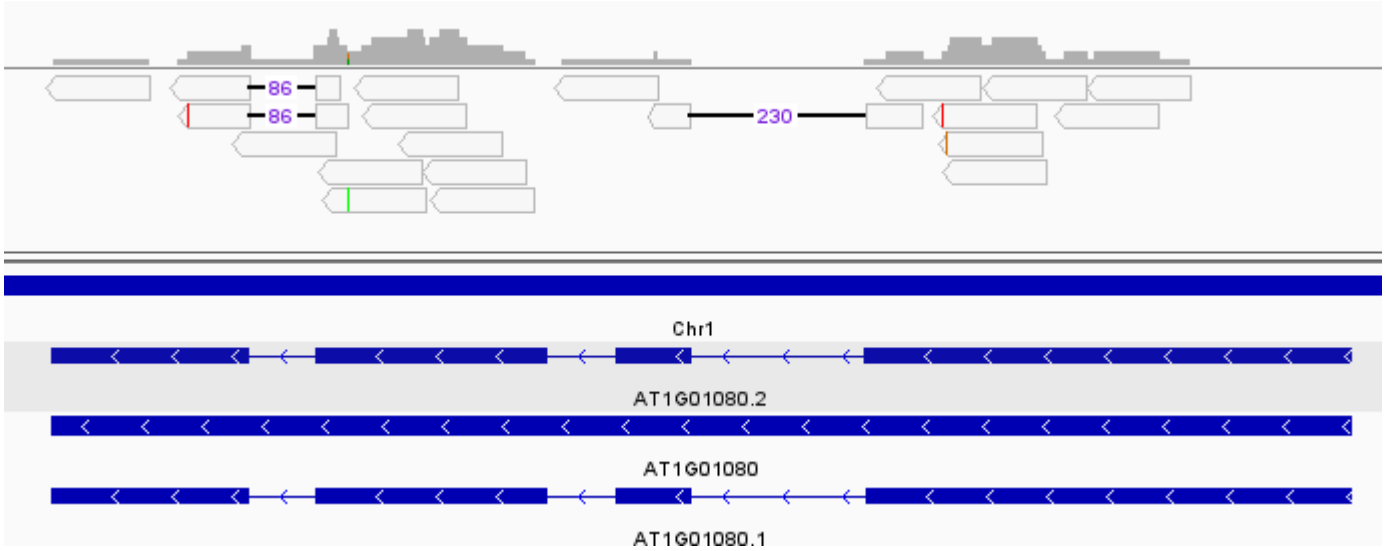
- Next slides are descriptions on what numbers were computed and saved in which files.
- They can be used not only for alternative splicing analyses.

rackj pipeline

- geneRPKM (by RPKMComputer)
  - a read count example

GeneID	Length	#Reads	RPKM	multi/ALL
AT1G01080	1.322	20	32.4825	0

20/(1.322x0.465) = 32.48

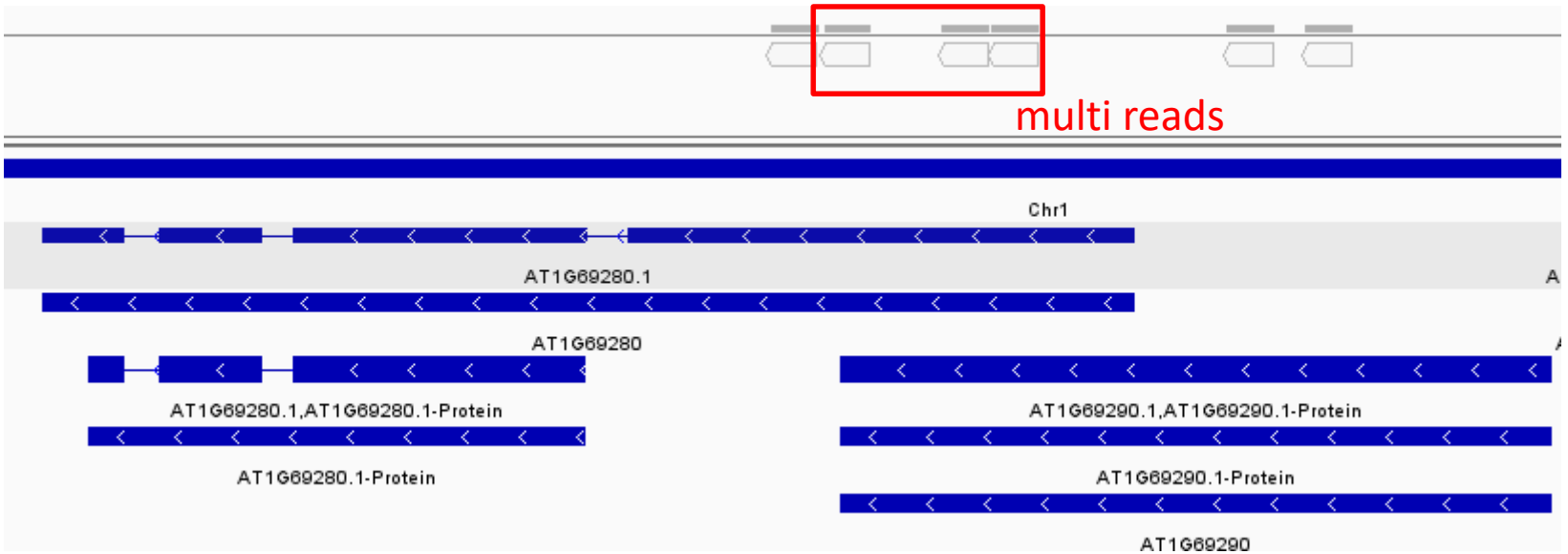


rackj pipeline

- geneRPKM (by RPKMComputer)
  - another read count example

GeneID	Length	#Reads	RPKM	multi/ALL	#uniq
AT1G69280	2.743	1.794721	1.404824	0.44281	1
AT1G69290	1.977	4.205279	4.567085	0.524407	2

1.79x(1-0.443)=1

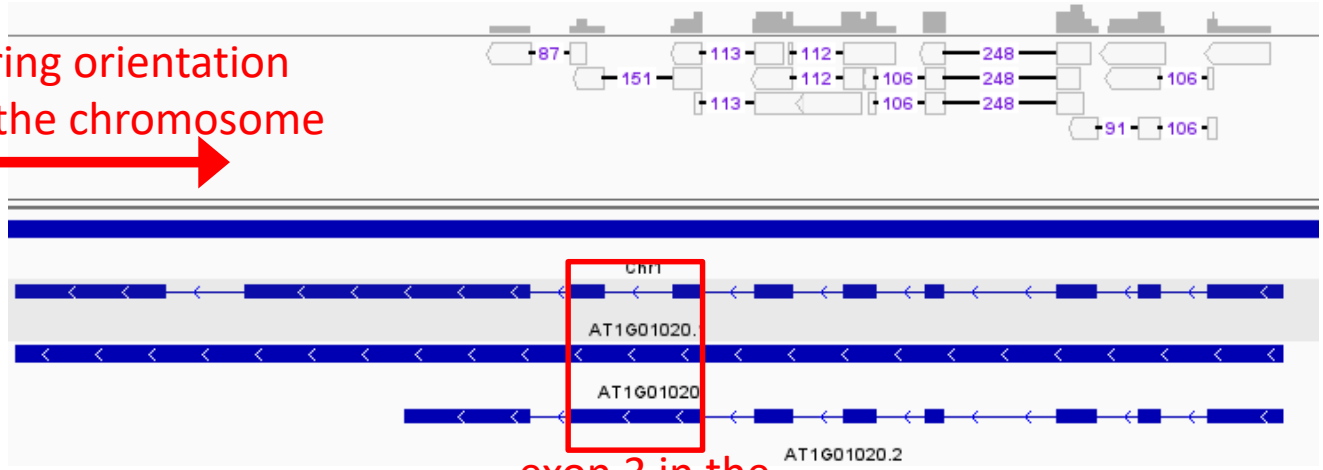


rackj pipeline

- exonCount/intronCount (RPKMComputer/ExonCounter -intronic true)

GeneID	exonNo	#Reads	exonLen	multi/ALL
AT1G01020	1	0	336	0
AT1G01020	2	1	633	0
AT1G01020	3	4	294	0
AT1G01020	4	4	86	0

Numbering orientation follows the chromosome

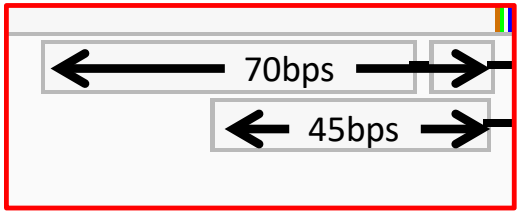


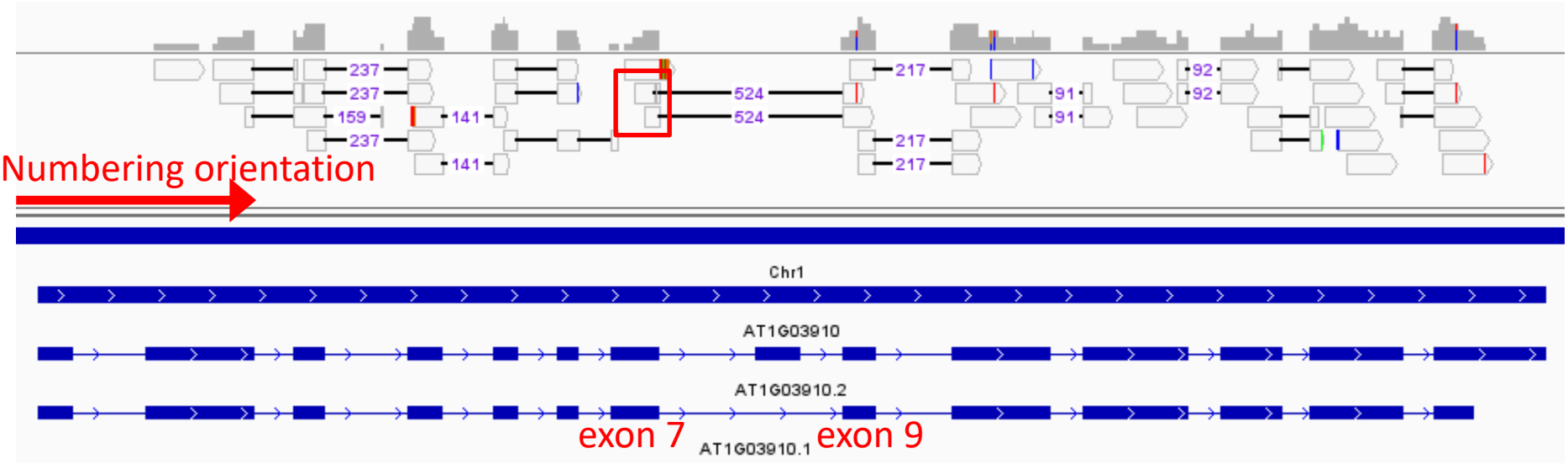
exon 3 in the merged model

rackj pipeline

- spliceCount (by RPKMComputer)
  - Jumping: skip some exon
  - Novel: this jumping is *novel*

GeneID	exonPair	#Reads	Jumping	Novel	splicingPosFreq
AT1G03910	7<=>9	2.0	V		{45=1, 70=1}





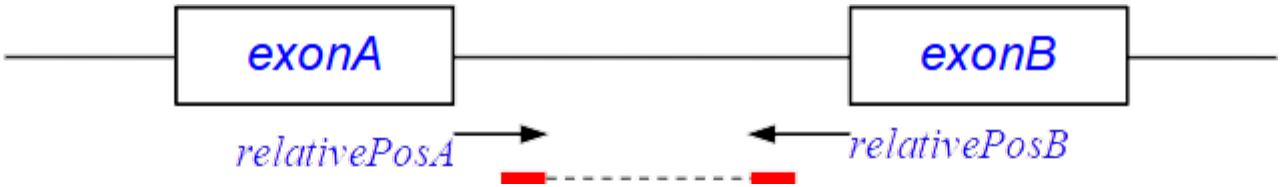
rackj pipeline

- fineSplice (by FineSpliceCounter)
  - Gives better resolution, in terms of splicing junctions, than spliceCount

GeneID	Splice	#Reads	Novel	splicingPosFreq
AT1G01650	10(-11)-11(0)	1	V	{71=1}

rackj pipeline

- fineSplice (by FineSpliceCounter)
  - We developed a system of notations to denote splicing junctions and called them *splicing patterns*
    - http://rackj.sourceforge.net/SpecialScripts/index.html#SeqGenAS
  - The most commonly used pattern is *exonA*(*relativePosA*)-*exonB*(*relativePosB*), for splicing junctions between two exons
    - A negative(positive) relative position means that the splicing site is inside(outside) the exon, and a zero relative position means that the splicing site agrees with that in the database.



rackj pipeline

- The unified notation for splicing patterns makes it possible to record various kinds of read counts
  - Thus various kinds of comparisons.

rackj pipeline

- Final comparison tables

Filename	Alternative splicing type	Merged sample?
SSDAs_*.xls	alternative donor/accepter	separate samples
SSESs_*.xls	alternative exon skipping	separate samples
SSIRs_*.xls	alternative intron retention	separate samples
SSDAm_*.xls	alternative donor/accepter	merged samples
SSESm_*.xls	alternative exon skipping	merged samples
SSIRm_*.xls	alternative intron retention	merged samples

rackj pipeline

- Comparisons of merged samples are based on aggregated numbers of reads of separate samples => higher statistical power
- Comparisons of separate samples taking cares of replications by applying T-TESTs

rackj pipeline

- In each table, look for columns headered by “P-value” or “TTEST” for P-values
  - The first P-value indicates *alternative* splicing
  - The second P-value (if any) indicates deviation from the constitutional form
    - the constitutional form: the form mostly expressed in the compared samples

Cases to be checked

- AT4G34150 for intron retention
- AT2G41100 for intron retention
- AT4G16695 for exon skipping
- AT1G23080 for alternative donor/accepter