Workflow for finding LncRNAs in *A. fumigatus* (strain A1163) – the UNION-PIPE

Candidate LncRNAs were extracted from 44 paired-end RNA-seq runs of *Aspergillus fumigatus* (strain A1163) exposed to various drug regimens from the Prof. M. Bromley's Group (University of Manchester).
Scheme makes four different StringTie merged assemblies from the same sample set using different initial StringTie assembly parameters and on all or half the data set, and then merges. This seemed the most conducive to LncRNA retrieval. Novel protein-coding genes were recovered simultaneously.

Workflow implementation capitalises on code and concepts used to uncover LncRNAs in *Candida* which used the merge of many runs to reveal the RNAPolII LncRNAs.

Hovhannisyan, H., Gabaldón, T. The long non-coding RNA landscape of Candida yeast pathogens. Nat Commun 12, 7317 (2021). https://doi.org/10.1038/s41467-021-27635-4
https://github.com/Gabaldonlab/lncRNAs

.

1. - Extract, clean, quality check, map and build StringTie assemblies of chosen reads.

Batch jobs run on the Manchester University Computational Shared Facility (**CSF**)

2. - Quality checks of mapped reads and StringTie transcriptome – Done on local machine using Bash commands to run small scripts and using conda as well as Jupyter notebooks for analysis (python 3.9.9).

3. - Making four different StringTie merges of StringTie assemblies for the pipeline (CSF).

4.- Transcripts processed for LncRNA assessment

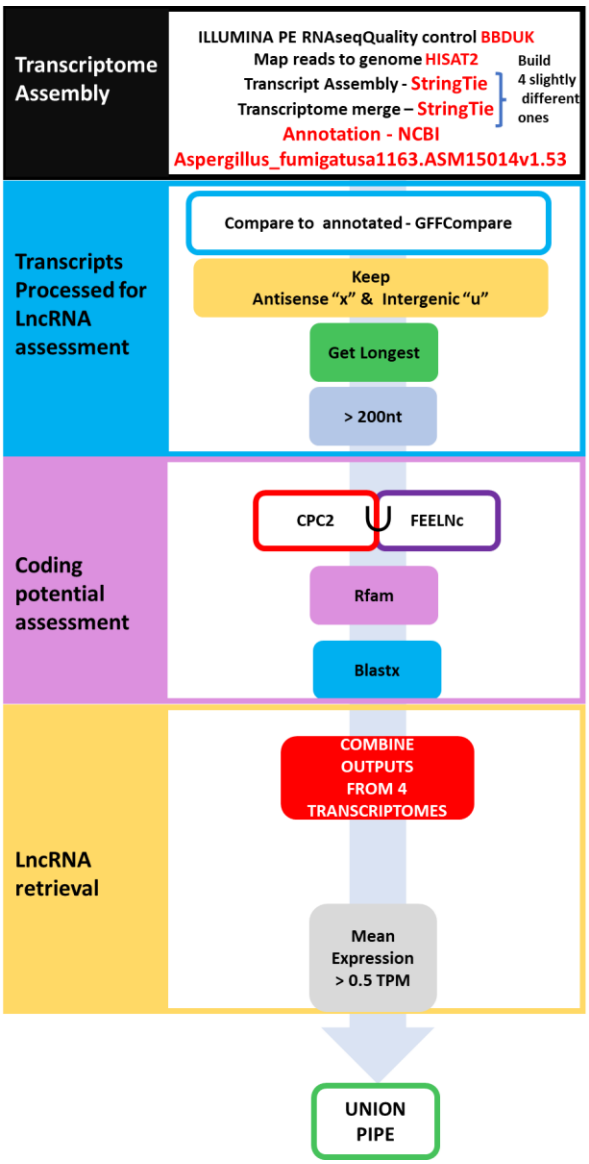
5. - Coding potential assessment

6. - LncRNA retrieval – pre TPM expression filter

7. - Novel Protein coding retrieval that does not overlap with newly found candidate LncRNAs

8. - Mean Expression level filter - TPM analysis

9. - Further processing of data: GC-Content, Length, Genome coverage, hit clustering, Exon number per transcript



1 Extract, clean, quality check, map and build StringTie assemblies of chosen reads. Manchester University Computational Shared Facility (CSF)

1.1 Trim and quality control using BBDUK https://github.com/BioInfoTools/BBMap/blob/master/sh/bbduk.sh

Requires:

accession.txt – list of run names

adapters.fa (https://github.com/BioInfoTools/BBMap/blob/master/resources/adapters.fa)

```
#!/bin/bash --login

#$ -cwd

#$ -t 1-44

#$ -pe smp.pe 8

#$ -N Marian_trimming_A1163

module load apps/binapps/anaconda3/2021.11

module load apps/bioinf

module load tools/env/proxy2

# Needs adapters.fa from BBmap, accession.txt runs to retrieve

sample=`sed -n "${SGE_TASK_ID}p" a1163_prefix.txt`

echo $sample

#Trim and quality check

conda activate bbmap_ML

bbduk.sh -Xmx1g in1=./new_fastq_files/"$sample"_R1_001.fastq
in2=./new_fastq_files/"$sample"_R2_001.fastq
out1="$sample"_trim_1.fastq out2="$sample"_trim_2.fastq
ref=adapters.fa ktrim=r k=23 hdist=1 mink=11 tpe tbo

bbduk.sh -Xmx1g in1="$sample"_trim_1.fastq
in2="$sample"_trim_2.fastq out1="$sample"_clean_1.fastq
out2="$sample"_clean_2.fastq qtrim=r trimq=10 maq=10 minlen=50

conda deactivate
```

1.2 Check quality with FASTQC and view each – if not correct readjust trim parameters

https://github.com/s-andrews/FastQC

Requires:

accession.txt – list of run names

```
#!/bin/bash --login
#$ -cwd
#$ -t 1-44
#$ -pe smp.pe 8
#$ -N Fastqc_newTrim_A1163
module load apps/binapps/anaconda3/2021.11
module load apps/bioinf
module load apps/fastqc/0.11.8/noarch
module load tools/env/proxy2
sample=`sed -n "${SGE_TASK_ID}p" a1163_prefix.txt`
echo $sample
fastqc -o fastqc_output_new_trim "$sample"_clean_1.fastq
fastqc -o fastqc_output_new_trim "$sample"_clean_2.fastq
```

1.3 Map onto genome with HISAT2 and BAM sort files with samtools and build a transcriptome with StringTie

http://daehwankimlab.github.io/hisat2/

https://github.com/samtools/samtools

https://github.com/gpertea/stringtie

KEEP ALL HiSAT2 GENERATED ERROR FILES – these contain map rates!

First make the HISAT2 index:

```
#!/bin/bash --login
#$ -cwd
#$ -N A1163_Hisat2_index
module load apps/binapps/hisat2/2.2.1
# build hisat2 index
hisat2-build
Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163_REF_INDEX
```

Now map all reads onto the genome and convert the SAM files to BAM files:

Requires:

accession.txt – list of run names

hisat2 index - A1163_REF_INDEX

```
#!/bin/bash --login
#$ -cwd
#$ -t 1-44
#$ -pe smp.pe 8
#$ -N Marian_map_A1163


module load apps/binapps/anaconda3/2021.11
module load apps/bioinf
module load apps/gcc/samtools/1.13
module load apps/binapps/hisat2/2.2.1
module load tools/env/proxy2
sample=`sed -n "${SGE_TASK_ID}p" a1163_prefix.txt`
echo $sample
PREFIX=$(echo $sample)
# HISAT2 map
hisat2 -p $NSLOTS --rg-id=$sample   -x A1163_REF_INDEX --dta --rna-strandness RF --max-intronlen 3100 --pen-noncansplice 1000000 --no-mixed \
--no-discordant -1 "$sample"_clean_1.fastq -2 "$sample"_clean_2.fastq -S "$sample".sam
# Convert Sam to Bam
samtools sort -@ $NSLOTS -o "$sample".bam "$sample".sam
```

HISAT2 parameters used:

-p 8 **(number of threads)**

--rg-id=AF293_1_01

-x AF293_REF_INDEX

--dta **(better for transcriptome assembly)**

--rna-strandness **RF (Equivalent to Salmon ISR)**

--max-intronlen **3100 (Biggest annotated intron is 3093)**

--pen-noncansplice 1000000 (standard is 12 but GTAG is standard in Aspergillus fumigatus Wang et al 2009, I recalculated > 99.6%(18623))

--no-mixed **(together with no-discordant, only when both sequences are matched are outputted)**

--no-discordant

```
-1 (reads1)

-2 (reads2)

-S ./AF293_1_01.sam (output SAM file)
```

1.4 Assemble reads into a transcriptome with StringTie using default parameters for quality checks.

Requires:

accession.txt

Aspergillus_fumigatus.ASM265v1.53.gtf This can be made using gffread

gffread 0.12.7 https://github.com/gpertea/gffread

```
gffread Aspergillus_fumigatusa1163.ASM15014v1.53.gff3 -T -o
Aspergillus_fumigatusa1163.ASM15014v1.53.gtf
```

```
#!/bin/bash --login

#$ -cwd

#$ -t 1-44

#$ -pe smp.pe 8

#$ -N Marian_Stringtie_default_A1163

module load apps/binapps/anaconda3/2021.11

module load apps/bioinf

module load tools/env/proxy2

mkdir st_A1163_default_12_7 -p

sample=`sed -n "${SGE_TASK_ID}p" a1163_prefix.txt`

echo $sample

conda activate stringtie_ml

stringtie --rf   -p $NSLOTS  -G
Aspergillus_fumigatusa1163.ASM15014v1.53.gtf -o
./st_A1163_default_12_7/A1163"$sample"default.gtf\

  -A ./st_A1163_default_12_7/A1163"$sample"default.tab
./A1163_bam/"$sample".bam

conda deactivate
```

---

2 Quality checks of mapped reads and StringTie transcriptome – Done on local machine using Bash commands and conda to run small scripts and then outputs calculated in a Jupyter notebook (python 3.9.9) - A1163_analysis_StringTie_Transcriptomes-GH.pdf

---

2.1 Number genes in longest predicted transcript - to see how much of a problem chimeric transcripts are.

2.1.a. Make a bed file of the genome annotation using gtf2bed (bedops 2.4.39 https://github.com/bedops/bedops)

```
gtf2bed Aspergillus_fumigatusa1163.ASM15014v1.53.gtf >
AF1163.v1.53.bed
```

2.1.b Extract the longest transcript for each locus into a GTF file. (Used this method as a couple of my StringTie assemblies were giving me problems with cgat's longest-transcript method)

Requires:

get_longest_pt1_v2.sh

get_longest.py

folder within it where the StringTie assembled gtfs are stored (st_A1163_default_12_7_GTF)

Outputs *gtflong.gtf files and gtf_list.txt

```
bash get_longest_pt1_v2.sh  st_A1163_default_12_7_GTF
```

2.1.c Converts the GTF to a Bed file

bedops(2.4.39)

Requires: *long.gtf files

Outputs: *long.bed files

get_longest_pt2.sh

```
bash get_longest_pt2.sh
```

2.1.d Extracts the number of annotated genes covered by a transcript on the same strand using bedtools intersect

Requires:
bedtools 2.30.0 https://github.com/arq5x/bedtools2
*long.bed files

AF1163.v1.53.bed

bash get_longest_A1163_pt3.sh

Outputs:

*long.bed.csv with overlap info

csv_bed_list.txt list of these files

```
bash get_longest_A1163_pt3.sh
```

2.1.e This section was analysed in Python Notebooks A1163_analysis_StringTie_Transcriptomes-GH.pdf

2.2 Percent sensitivity and precision at base level compared to annotation.

2.2.a extract information on how well each individual StringTie assembly does per base sensitivity, precision compared to the annotation using GFFCompare.

Requires:

gffcompare 0.11.2 https://github.com/gpertea/gffcompare

gtf_list.txt generated section:2.1.b

Aspergillus_fumigatusa1163.ASM15014v1.53.gtf

(gff_compare_polycistronc_A1163_ST.sh script will have to be altered very slightly as a little too tailored for my directory name)

Outputs: trial_output.txt

```
bash gff_compare_polycistronc_A1163.sh
```

2.2.b The information was then extracted in python within Jupyter notebooks – see **A1163_analysis_StringTie_Transcriptomes-GH.pdf**

2.3 Mapping of Intron overlaps to check that transcripts on opposite strands do not have a lot of matching. For intron overlaps – all mapped introns in the StringTie annotations are used

Make a temporary copy of all the programmes to a new folder within the folder you have been working in and change directory to that folder.

Programmes:

bed_intersect_intron.sh

get_intron_beds.A1163.sh

get_intron_overlap_stats.sh

hisat2_extract_splice_sites.py (from hisat2)

intron_bed_making_get_stats_A1163.py

intron_bed_making_just_beds1.py

2.3.a Extracts all the intron sites using hisat2_extract_splice_sites.py (from hisat2) and outputs the intron starts and stops as two files based on strand. It also generates a list of files bedMake_list.txt which it uses by running intron_bed_making_just beds1.py to convert them to bed like files..

Requires:

gtf_list.txt of StringTie GTFs

hisat2_extract_splice_sites.py
(https://github.com/DaehwanKimLab/hisat2/blob/master/hisat2_extract_splice_sites.py)

get_intron_beds_A1163.sh

intron_bed_making_just_beds1.py

outputs: to_comp_introns2.txt

```
bash get_intron_beds_A1163.sh ./../gtf_list.txt
```

2.3.b The overlap regions of introns on the two strands are then extracted *via* bedtools. Using bed_intersect_intron.sh and the to_comp_introns2.txt list generated previously extracts overlaps between introns on the two different strands

Requires:

bedtools 2.30.0

bed_intersect_intron.sh

```
bash bed_intersect_intron.sh to_comp_introns2.txt
```

2.3.c Calculate the stats of the overlap

Requires:

intron_bed_making_get_stats_A1163.py

get_intron_overlap_stats.sh

outputs intron_overlap.csv

```
python intron_bed_making_get_stats_A1163.py
```

2.3.d The information was then extracted in python within Jupyter notebooks – see **A1163_analysis_StringTie_Transcriptomes-GH.pdf**

2.4 Number of mapped reads per run and map rate can be extracted from the HISAT2 error message

3 Making four different StringTie merges of StringTie assemblies for the pipeline (CSF).

Types of StringTie merges:

Name	Samples	Assembly gap
E	all	50
F	all	10
G	half	50
H	half	10

Samples were then split into a high-precision half of 22 samples as this correlated the best with reducing the number of polycistronic messages. Carried out default StringTie assembly and merge on full set and half set. Also restricted the size of the "gap" allowed between non-overlapping reads from default of 50 to 10 in the transcriptome assembly for both groups and then merged using default values. For assembly of transcriptomes using default values can use what made before in §1.4. In addition, repeat this with -g 10 parameter to the assembly. The assemblies were then merged into single transcriptomes and their quality could be

assessed as for the single StringTie assemblies as well as viewing them on the Integrative Genomics Viewer.

Requires: Aspergillus_fumigatus.ASM265v1.53.gtf

```
#!/bin/bash --login

#$ -cwd

#$ -pe smp.pe 4

#$ -N Marian_Stringtie_merge_A1163_default_all

module load apps/binapps/anaconda3/2021.11

module load apps/bioinf

module load tools/env/proxy2

cd "/net/scratch2/s99384ml/A1163/st_A1163_default_12_7/"

ls -1 *.gtf > 0839gtf_list.txt

conda activate stringtie_ml

stringtie --merge -G
"./../Aspergillus_fumigatusa1163.ASM15014v1.53.gtf" -p $NSLOTS   -o
"./../A1163_default_all_merge.gtf" 0839gtf_list.txt

conda deactivate
```

4 Transcripts processed for LncRNA assessment

 # STEP 4.1 - compares StringTie transcriptome to annotated Requires gffcompare 0.11.2

```
conda activate GFF_utils

while read p; do

 echo "$p"

# get prefix

 prefix=$(echo "$p"|awk -F "A1163_stringtie_GTFS/" '{print $2}'|awk
-F "_merge" '{print $1}')

echo "$prefix"

mkdir "$prefix"

echo "$prefix" >> prefix.txt

cd  "$prefix"

gffcompare -V ../"$p" -o "$prefix"_merged_compared.gtf -r
../Aspergillus_fumigatusa1163.ASM15014v1.53.gff3

cd ..

done <gtf_list.txt
```

```
conda deactivate
```

#STEP 4.2 - Highlight unannotated transcripts

Requires: non_cod.py https://github.com/Gabaldonlab/lncRNAs

```
while read p; do
  echo "$p"
cd "$p"
python ../non_cod.py "$p"_merged_compared.gtf.annotated.gtf
"$p"_unknown_and_antisense_ids.gtf
cd ..
done <prefix.txt
```

#STEP 4.3 - pulls out longest transcript of new loci in a GTF requires CGAT - Computational Genomics Analysis Tools https://cgat-apps.readthedocs.io/en/latest/

```
conda activate cgat_ml2
while read p; do
  echo "$p"
cd "$p"
cgat gtf2gtf  --method=filter --filter-method=longest-transcript -I
"$p"_unknown_and_antisense_ids.gtf >
"$p"_unknown_and_antisense_longest_transcripts.gtf
cd ..
done <prefix.txt
conda deactivate
```

#STEP 4.4 - Makes longest transcripts/locus in a fasta file requires gffread 0.12.7

```
conda activate GFF_utils
while read p; do
  echo "$p"
cd "$p"
gffread -w "$p"_unknown_and_antisense_transcripts.fasta -W -F -g
../Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
"$p"_unknown_and_antisense_longest_transcripts.gtf
cd ..
done <prefix.txt
conda deactivate
```

```
# Records initial number detected _ count_1

while read p; do

  echo "$p"

cd "$p"

echo "$p">> ../count1.txt

grep ">" "$p"_unknown_and_antisense_transcripts.fasta|wc -l>>
../count1.txt

cd ..

done <prefix.txt
```

#STEP 4.5 - selects those over 200 nt

Requires: select_longer_200.py https://github.com/Gabaldonlab/lncRNAs

```
while read p; do

  echo "$p"

cd "$p"

echo "$p">> ../count2.txt

python ../select_longer_200.py
"$p"_unknown_and_antisense_transcripts.fasta
"$p"_unknown_and_antisense_transcripts_longer200.fasta

# record number

grep ">" "$p"_unknown_and_antisense_transcripts_longer200.fasta|wc -
l>> ../count2.txt

cd ..

done <prefix.txt
```

#STEP 4.6 - Removes those with ambiguous sequence requires remove_seqs_with_amb_nucl.py https://github.com/Gabaldonlab/lncRNAs

Aim: to get rid of transcripts that may have been created due to poor sequencing

Needs bio.alphabet so requires earlier biopython such as biopython=1.68 (had to do it with pip in base environment）

```
while read p; do

echo "$p"

cd "$p"

echo "$p">> ../count3B.txt

python ../remove_seqs_with_amb_nucl.py
"$p"_unknown_and_antisense_transcripts_longer200.fasta
```

```
"$p"_unknown_and_antisense_transcripts_longer200_no_amb_nucl.fasta >
"$p"_removed_transcripts_with_amb_nuclt.txt

##record the number

grep ">"
"$p"_unknown_and_antisense_transcripts_longer200_no_amb_nucl.fasta|w
c -l>> ../count3B.txt

cd ..

done <prefix.txt
```

5 Coding potential assessment

#STEP 5.1 - Finds those with low protein coding potential using CPC2

Download CPC2_standalone_python3 v1.0.1 https://github.com/gao-lab/CPC2_standalone

http://cpc2.gao-lab.org/download.php (A bit fiddly to install compared to conda packages)

```
while read p; do

  echo "$p"

cd "$p"

python /home/marian-linux/CPC2-beta/bin/CPC2.py -i
"$p"_unknown_and_antisense_transcripts_longer200.fasta -o
"$p"_CP2_results.tab

cd ..

done <prefix.txt
```

#STEP 5.2 - Finds those with low protein coding potential using FEELNC use on UseGalaxy.org

```
# To get all wanted files to same folder

mkdir A1163_galaxy -p

while read p; do

echo "$p"

cd "$p"

cp "$p"_unknown_and_antisense_longest_transcripts.gtf
../A1163_galaxy

cd ..

done <prefix.txt
```

```
 For GALAXY

"$p"_unknown_and_antisense_longest_transcripts.gtf
```

```
Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa

Aspergillus_fumigatusa1163.ASM15014v1.53.gtf

# save each as: "prefix"_FEELNC.txt
```

#STEP 5.3 get rid of known RNA genes – Actually want to allow Afu-309 and Afu-182 through, this is done at step 5.6.

infernal 1.1.2 https://github.com/EddyRivasLab/infernal

Also requires in same folder:

Rfam.cm.i1p, Rfam.cm.i1m,Rfam.cm.i1i,Rfam.cm.i1f,Rfam.clanin

```
conda activate infernal

while read p; do

  echo "$p"

cmscan --nohmmonly --rfam --cut_ga --fmt 2 --oclan --oskip --clanin
Rfam.clanin -o ./"$p"/"$p"_lncRNAs.cmscan.out --tblout
./"$p"/cmscan_"$p"_lncRNAs.tblout Rfam.cm
./"$p"/"$p"_unknown_and_antisense_transcripts_longer200_no_amb_nucl.
fasta

done <prefix.txt

conda deactivate
```

#STEP 5.4 Allow any transcript that has passed either CPC2 or FEELNC as noncoding to pass through

Requires: CPC2andFEELNC_UNION.py

#As also had a pipe trialling what it was like if both had to be true, files were renamed containing "UNION" and the prefix_UNION.txt was created in place of prefix.txt as shown below:

A1163_default_all_UNION

A1163_default_HP_UNION

A1163_g10_all_UNION

A1163_g10_HP_UNION

```
#Need to do union CPC2 and FEELNC

while read p; do

  echo "$p"

cd "$p"

python ../CPC2andFEELNC_UNION.py -i "$p"_CP2_results.tab.txt -g
"$p"_noncoding_ids_feelnc.txt> AF293_Both_NC.txt

cd ..

done <prefix_UNION.txt
```

#STEP 5.5 Make a GFF as easier to sort

```
while read p; do
  echo "$p"
cd "$p"
conda activate GFF_utils
gffread -E "$p"_unknown_and_antisense_longest_transcripts.gtf -o-  > "$p"_unknown_and_antisense_longest_transcripts.gff3
cd ..
done <prefix_UNION.txt
conda deactivate
```

#STEP 5.6 – remove Rfam hits and only keep noncoding. Exception is that want to make sure if there is anything at LncRNA loci 5_ureB_sRNA, Afu_182, Afu_309 that they are let through

Requires: sortgtf_no_coding_noRFAM_23_7.py

Outputs: a gff of lncrna transcripts LncRNA_refined.gff

```
while read p; do
  echo "$p"
cd "$p"
python ../sortgtf_no_coding_noRFAM_23_7.py --i "$p"
cd ..
done <prefix_UNION.txt
```

#STEP 5.7 – Make transcripts for blast

```
conda activate GFF_utils
while read p; do
  echo "$p"
cd "$p"
gffread -w "$p"_LncRNA_refined.fasta -W -F -g ../Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa "$p"_LncRNA_refined.gff
cd ..
done <prefix_UNION.txt
conda deactivate
```

#STEP 5.8 – now need to grab all relevant files for BLAST and a list of files we want to search against

```
mkdir for_CSF_BLAST_A1163_UNION
while read p; do
  echo "$p"
cd "$p"
cp "$p"_LncRNA_refined.fasta ../for_CSF_BLAST_A1163_UNION
cd ..
done <prefix_UNION.txt
ls -1  ./for_CSF_BLAST_A1163_UNION/*.*> A1163_blast_list_UNION.txt
```

#STEP 5.9 Do BlastX against all Aspergillus reference Proteins Done on Manchester University Computational Shared Facility (CSF)

https://github.com/ncbi/blast_plus_docs

(Takes a lot of computation time)

Aspergillus reference proteins were downloaded from NCBI 4/7/22

https://www.ncbi.nlm.nih.gov/protein?term=%22Aspergillus%22%5BOrganism%5D%20AND%20refseq%5Bfilter%5D%20&cmd=DetailsSearch

First must make a database and then you can search it

```
#!/bin/bash --login
#$ -cwd
#$ -pe smp.pe 8
#$ -t 1-4
#$ -N BLAST_PREP_For_A1163_UNION
module load apps/bioinf
module load tools/env/proxy2
module load apps/binapps/blast/2.9.0
sample=`sed -n "${SGE_TASK_ID}p" A1163_blast_list_UNION.txt`
echo $sample
makeblastdb -in Aspergillus.fasta -parse_seqids -dbtype prot
blastx -query "$sample" -db Aspergillus.fasta  -outfmt 10 -evalue
1e-3  -max_hsps 2 -out "$sample"_BLAST_HITS.csv
```

#STEP 5.10 Removing_BLAST_HITS

Download, and rename

Requires: Removing_BLAST_HITS_for_new_filter_revamped_4_8.py

This algorithm contrasts how much homology there is on the sense and antisense strands to Aspergillus predicted protein coding genes to assess the nature of the candidate LncRNAs. A threshold of more than one hit, and one with more than 90 bp overlap is also imposed to limit noise.

```
while read p; do   echo "$p"; python
Removing_BLAST_HITS_for_new_filter_revamped_4_8.py -i "$p"; done
<prefix_UNION.txt>>trial.txt

#then check u and x

conda activate GFF_utils

while read p; do

  echo "$p"

cd "$p"

gffcompare "$p"_LncRNA_refined_blast_removed.gff -o
for_u_x_class_"$p".stats -r
./../Aspergillus_fumigatusa1163.ASM15014v1.53.gff3

cd ..

done <prefix_UNION.txt

conda deactivate
```

6 LncRNA retrieval – pre TPM expression filter

#STEP 6.1 – Now merge on longest - careful using the same names as before

```
# first give genes their unique names

cat
A1163_g10_HP_UNION/A1163_g10_HP_UNION_LncRNA_refined_blast_removed.g
ff | sed -r 's/'MSTRG'/LncT_H_/g'>
A1163_g10_HP_UNION/A1163_g10_HP_UNION_LncRNA_refined_blast_removed_r
enamed.gff

cat
A1163_g10_all_UNION/A1163_g10_all_UNION_LncRNA_refined_blast_removed
.gff | sed -r 's/'MSTRG'/LncT_F_/g'>
A1163_g10_all_UNION/A1163_g10_all_UNION_LncRNA_refined_blast_removed
_renamed.gff

cat
A1163_default_HP_UNION/A1163_default_HP_UNION_LncRNA_refined_blast_r
emoved.gff | sed -r 's/'MSTRG'/LncT_G_/g'>
A1163_default_HP_UNION/A1163_default_HP_UNION_LncRNA_refined_blast_r
emoved_renamed.gff

cat
A1163_default_all_UNION/A1163_default_all_UNION_LncRNA_refined_blast
_removed.gff | sed -r 's/'MSTRG'/LncT_E_/g'>
```

```
A1163_default_all_UNION/A1163_default_all_UNION_LncRNA_refined_blast
_removed_renamed.gff
```

#STEP 6.2 – Combine outputs of all preps together

```
cat
A1163_g10_HP_UNION/A1163_g10_HP_UNION_LncRNA_refined_blast_removed_r
enamed.gff
A1163_g10_all_UNION/A1163_g10_all_UNION_LncRNA_refined_blast_removed
_renamed.gff
A1163_default_HP_UNION/A1163_default_HP_UNION_LncRNA_refined_blast_r
emoved_renamed.gff
A1163_default_all_UNION/A1163_default_all_UNION_LncRNA_refined_blast
_removed_renamed.gff > merge_A1163_UNION_L_step1.gff
```

#STEP 6.3 - now get rid of "gene" from here

```
cat  merge_A1163_UNION_L_step1.gff | grep "ID=gene" -v>
merge_A1163_UNION_L_step2.gff
```

#STEP 6.4 - compare to annotated transcriptome

```
conda activate GFF_utils
```

```
gffcompare -V merge_A1163_UNION_L_step2.gff -o
merge_A1163_UNION_L_step2.stats -r
./Aspergillus_fumigatusa1163.ASM15014v1.53.gff3
```

```
conda deactivate
```

#STEP 6.5 - need to remove duplicate features

```
conda activate cgat_ml2
```

```
cgat gtf2gtf --method=remove-duplicates --duplicate-feature
"transcript" -I  merge_A1163_UNION_L_step2.annotated.gtf >
merge_A1163_UNION_L_step3.gtf
```

```
conda deactivate
```

#STEP 6.6 - need to pull out longest transcript

cgat did not work for me so did it in python

requires:

pull out longest transcript

collapse_to_longest.py

outputs: merge_A1163_UNION_L_step3.gtflongest_.gtf

```
python collapse_to_longest.py -i  merge_A1163_UNION_L_step3.gtf
```

#STEP 6.7 – Check how it went

```
conda activate GFF_utils

gffcompare -V merge_A1163_UNION_L_step3.gtflongest_.gtf

-o  merge_A1163_UNION_L_step4_6_8.longest-N.stats -r
./Aspergillus_fumigatusa1163.ASM15014v1.53.gff3

conda deactivate
```

#STEP 6.8 – change XLOCNAMES

```
cat merge_A1163_UNION_L_step4.longest_.gtf | sed -r
's/'XLOC_'/LncA1163_T_/g'>merge_A1163_UNION_L_step4.longest_b.gtf
```

#STEP 6.9 – Extract X and Us non_cod.py non_cod.py https://github.com/Gabaldonlab/lncRNAs

non_codU.py non_codX.py are simple variations of non_cod.py that just take one of the classes

```
python non_cod.py merge_A1163_UNION_L_step4.longest_b.gtf
A1163_ML_UX_LncRNA_T_6_8.gtf

python non_codU.py merge_A1163_UNION_L_step4.longest_b.gtf
A1163_ML_U_LncRNA_T_6_8.gtf

python non_codX.py merge_A1163_UNION_L_step4.longest_b.gtf
A1163_ML_X_LncRNA_T_6_8.gtf
```

#STEP 6.10 – Now get transcripts in fasta

```
conda activate GFF_utils

gffread -w A1163_ML_UX_LncRNA_T_6_8.fa -W -F -g
Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163_ML_UX_LncRNA_T_6_8.gtf

gffread -w A1163_ML_X_LncRNA_T_6_8.fa -W -F -g
Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163_ML_X_LncRNA_T_6_8.gtf

gffread -w A1163_ML_U_LncRNA_T_6_8.fa -W -F -g
Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163_ML_U_LncRNA_T_6_8.gtf

# fastx_toolkit  0.0.14

fasta_formatter -i A1163_ML_UX_LncRNA_T_6_8.fa -o
formatted_A1163_ML_UX_LncRNA_T_6_8.fa

conda deactivate
```

7 Novel potential Protein coding retrieval that does not overlap with newly found candidate LncRNAs

#STEP 7.1 Do for each transcriptome – individually

Example shown: A1163_default_all (all preps default StringTie methods)

```
# Ones longer than 200nt without ambiguous sequence and make sure
they are unique

cat
A1163_default_all_UNION_unknown_and_antisense_transcripts_longer200_
no_amb_nucl.fasta|grep ">"|awk -F ">" '{print
$2}'>A1163_default_all_UNION_Kept_after_ambiguous.txt

#Then get the ones thought to be non-coding by CPC2 and/or FEELnc

cat
A1163_default_all_UNION_Kept_after_ambiguous.txt|sort|uniq>A1163_def
ault_all_UNION_Kept_after_ambiguous_sorted.txt

cat AF293_Both_NC.txt|sort|uniq>A_Both_NC_sorted.txt

# then see ones in first file that are not in second file

comm -23 A1163_default_all_UNION_Kept_after_ambiguous_sorted.txt
A_Both_NC_sorted.txt>A1163_default_all_UNION_dropped_FEELNC_CPC2.txt

# now combine blast hits and these files into one file

cat A1163_default_all_UNION_dropped_FEELNC_CPC2.txt
A1163_default_all_UNIONblast_removed_NF.txt >
A1163_default_all_UNION_BLAST_FEELNC_CPC2_DISCARDS.txt

#Remove Rfam hits from this as do not want RFAM stuff in here,
except Afu-182 Afu_309 5_ureB_sRNA because it will be interesting to
see it

cat cmscan_A1163_default_all_UNION_lncRNAs.tblout| grep Afu_182 -
v|grep Afu_309 -v| grep 5_ureB_sRNA -v|awk $6 '{print
$4}'>PREP_A_RFAM_HITS.txt

# open file get rid of rubbish

tail -n +3 PREP_A_RFAM_HITS.txt| head -n -
10>A1163_default_all_UNION_RFAM_HITS_G.txt

# Now can make final list of potential protein coding

cat
A1163_default_all_UNION_RFAM_HITS_G.txt|sort|uniq>A1163_default_all_
UNION_RFAM_HITS_G_sorted.txt

cat
A1163_default_all_UNION_BLAST_FEELNC_CPC2_DISCARDS.txt|sort|uniq>A11
63_default_all_UNION_BLAST_FEELNC_CPC2_DISCARDS_sorted.txt
```

```
comm -23
A1163_default_all_UNION_BLAST_FEELNC_CPC2_DISCARDS_sorted.txt
A1163_default_all_UNION_RFAM_HITS_G_sorted.txt>A1163_default_all_UNI
ON_dropped_FEELNC_CPC2_BLAST_notRFAM_DISCARDS.txt
```

#STEP 7.2- Makes the GTF of all potential novel coding genes

Requires a prefix

Outputs

A1163_default_all_UNION_Pot_Proteins_removed.gtf

```
python Making_Final_bits_for_pot_protein_coding.py -p
A1163_default_all_UNION
```

#STEP 7.3 - make new directory and go to it put copies inside:

A1163_default_all_UNION_Pot_Proteins_removed.gtf etc

#STEP 7.4 - first give genes their unique names

```
cat A1163_default_all_UNION_Pot_Proteins_removed.gtf | sed -r
's/'MSTRG'/PPT_E_/g'>
A1163_default_all_UNION_POT_Proteins_6_8_removed_renamed.gtf
```

```
cat A1163_g10_all_UNION_Pot_Proteins_removed.gtf | sed -r
's/'MSTRG'/PPT_F_/g'>
A1163_g10_all_UNION_POT_Proteins_6_8_removed_renamed.gtf
```

```
cat A1163_default_HP_UNION_Pot_Proteins_removed.gtf | sed -r
's/'MSTRG'/PPT_G_/g'>
A1163_default_HP_UNION_POT_Proteins_6_8_removed_renamed.gtf
```

```
cat A1163_g10_HP_UNION_Pot_Proteins_removed.gtf | sed -r
's/'MSTRG'/PPT_H_/g'>
A1163_g10_HP_UNION_POT_Proteins_6_8_removed_renamed.gtf
```

#STEP 7.5 - merge to one

```
cat A1163_default_all_UNION_POT_Proteins_6_8_removed_renamed.gtf
A1163_g10_all_UNION_POT_Proteins_6_8_removed_renamed.gtf
A1163_default_HP_UNION_POT_Proteins_6_8_removed_renamed.gtf
A1163_g10_HP_UNION_POT_Proteins_6_8_removed_renamed.gtf>merge_A1163P
_UNION.gtf
```

#STEP 7.6 - now get rid of gene from here

```
cat merge_A1163P_UNION.gtf | grep "ID=gene" -v>
merge_A1163P_UNION2.gtf
```

#STEP 7.7 - compare to annotated transcriptome

```
conda activate GFF_utils

gffcompare -V merge_A1163P_UNION2.gtf -o merge_A1163P_UNION2.stats -
r ./../Aspergillus_fumigatusa1163.ASM15014v1.53.gtf

conda deactivate
```

#STEP 7.8 - remove duplicate features

```
conda activate cgat_ml2

cgat gtf2gtf --method=remove-duplicates --duplicate-feature
"transcript" -I  merge_A1163P_UNION2.annotated.gtf >
merge_A1163P_UNION2.annotated2.gtf

conda deactivate
```

#STEP 7.9 - pull out longest-transcript

```
conda activate cgat_ml2

cgat gtf2gtf  --method=filter --filter-method=longest-transcript -I
merge_A1163P_UNION2.annotated2.gtf>
merge_A1163P_UNION2.annotated2_longest.gtf

conda deactivate
```

#STEP 7.10 – Now need to compare to the lncRNA final set to make sure they do not overlap and pull out x and u classes

```
conda activate GFF_utils

gffcompare -V merge_A1163P_UNION2.annotated2.gtf -o
A1163P_UNION_LongestvslncRNA.stats -r A1163_ML_UX_LncRNA_T_6_8.gtf

conda deactivate

# Need to pull out x and u classes so not overlapping with LncRNAs

python ./../non_cod.py A1163P_UNION_LongestvslncRNA.annotated.gtf
A1163P_UNION_LongestvslncRNA.annotated_UX.gtf
```

#STEP 7.11 – now get rid of gene from here

```
cat A1163P_UNION_LongestvslncRNA.annotated_UX.gtf| grep "ID=gene" -
v> A1163P_UNION_LongestvslncRNA.annotated_UX2.gtf
```

#STEP 7.12 – compare to annotated transcriptome

```
conda activate GFF_utils
```

```
gffcompare -V A1163P_UNION_LongestvslncRNA.annotated_UX2.gtf -o
A1163P_UNION_Longest_clean.stats -r
./../Aspergillus_fumigatusa1163.ASM15014v1.53.gtf

conda deactivate
```

#STEP 7.13 – need to remove duplicate features

```
conda activate cgat_ml2cgat gtf2gtf --method=remove-duplicates --
duplicate-feature "transcript" -I
A1163P_UNION_Longest_clean.annotated.gtf >
A1163P_UNION_Longest_clean.annotated2.gtf

conda deactivate
```

#STEP 7.14 – pull out longest transcript

Requires: collapse_to_longest.py

Outputs: A1163P_UNION_Longest_clean.annotated2. gtflongest_.gtf

```
python collapse_to_longest.py -i
A1163P_UNION_Longest_clean.annotated2.gtf
```

#STEP 7.15 – Now compare to see how it went

```
conda activate GFF_utils

gffcompare -V A1163P_UNION_Longest_clean.annotated2. gtflongest_.gtf
-o  A1163P_UNION_Longest_clean_FINAL.stats -r
./../Aspergillus_fumigatusa1163.ASM15014v1.53.gtf

conda deactivate
```

#STEP 7.16 – change XLOCNAMES

```
A1163P_UNION_Longest_clean_FINAL.annotated.gtf | sed -r
's/'XLOC_'/PPA1163T_/g'>A1163P_UNION_Longest_clean_FINAL2.annotated.
gtf
```

#STEP 7.17 – Now need to extract X and Us from this

py non_cod.py https://github.com/Gabaldonlab/lncRNAs

non_codU.py non_codX.py are simple variations of non_cod.py that just take one of the classes

```
python ./../non_cod.py
A1163P_UNION_Longest_clean_FINAL2.annotated.gtf A1163PT_UX_6_8.gtf

python ./../non_codU.py
A1163P_UNION_Longest_clean_FINAL2.annotated.gtf A1163PT_U_6_8.gtf

python ./../non_codX.py
A1163P_UNION_Longest_clean_FINAL2.annotated.gtf A1163PT_X_6_8.gtf
```

#STEP 7.18 – Now get transcripts in fasta

```
conda activate GFF_utils
```

```
gffread -w A1163PT_UX_6_8.fasta -W -F -g
./../Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163PT_UX_6_8.gtf

gffread -w A1163PT_X_6_8.fasta -W -F -g
./../Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163PT_X_6_8.gtf

gffread -w A1163PT_U_6_8.fasta -W -F -g
./../Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163PT_U_6_8.gtf

fasta_formatter -i A1163PT_UX_6_8.fasta -o
formatted_A1163PT_UX_6_8.fasta

conda deactivate
```

8 Mean Expression level filter -TPM analysis

#STEP 8.1 - Making a SAF File to map read counts onto with featureCounts
https://github.com/ShiLab-Bioinformatics/subread

Need to incorporate LncRNA and new potential protein coding genes in a saf file with the regions covered by protein coding genes in the annotation so that can use featureCounts

```
# For LncRNA

cat  A1163_ML_UX_LncRNA_T_6_8.gtf| awk '{if ($3=="transcript") print
$0}'>transcript_A1163_ML_UX_LncRNA_T_6_8.gtf

# Make into a bed file

conda activate GFF_utils

gtf2bed <transcript_A1163_ML_UX_LncRNA_T_6_8.gtf>
transcript_A1163_ML_UX_LncRNA_T_6_8.bed

# For Potential protein coding

cat A1163PT_UX_6_8.gtf| awk '{if ($3=="transcript") print
$0}'>transcript_A1163PT_UX_6_8.gtf

gtf2bed <transcript_A1163PT_UX_6_8.gtf>
transcript_A1163PT_UX_6_8.bed

# For A1163

#take protein-coding genes out of A1163 file then make a bed

cat Aspergillus_fumigatusa1163.ASM15014v1.53.gff3| awk '{if
($3=="gene") print $0}'>gene_A1163.gff3

gff2bed <gene_A1163.gff3> transcript_gene_A1163.bed

conda deactivate
```

```
# merge the bed files and sort

cat transcript_A1163_ML_UX_LncRNA_T_6_8.bed
transcript_A1163PT_UX_6_8.bed transcript_gene_A1163.bed >
merged_A1163_6_8_T.bed

# Now sort this bed file

sort -V -k1,1 -k2,2 merged_A1163_6_8_T.bed
>superbedmerged_A1163_6_8_T.bed

#then make two  saf files one for FeatureCount one for sorting
afterwards

awk 'OFS="\t" {print $1"."$2"."$3, $1, $2, $3, $4}'
superbedmerged_A1163_6_8_T.bed> superbedmerged_A1163_6_8_T_B.saf

awk 'OFS="\t" {print $1"."$2"."$3, $1, $2, $3, $5}'
superbedmerged_A1163_6_8_T.bed> superbedmerged_A1163_6_8_T_A.saf
```

#STEP 8.2 - Do feature count Done on Manchester University Computational Shared Facility (CSF).

```
!/bin/bash --login

#$ -cwd

#$ -pe smp.pe 8

#$ -N Marian_Feature_count_A1163_28_7

module load apps/binapps/anaconda3/2021.11

module load tools/env/proxy2

module load apps/gcc/samtools/1.13

conda activate subread_ml2

featureCounts -F SAF -p -T 2 -s 2 -a merged_UNION_A1163_28_7_B.saf
-o A1163_UNION_counts_28_7.txt 5-Fluorocytosine_x0_5-2_S59.bam 5-
Fluorocytosine_x1-1_S61.bam 5-Fluorocytosine_x1-3_S63.bam 5-
Fluorocytosine_x2-1_S64.bam Dodin_x0_5-1_S4.bam Dodin_x1-3_S9.bam
Dodin_x4-3_S15.bam FG10_CCGTCC_L001.bam FG11_GTCCGC_L001.bam
FG12_GTGAAA_L001.bam FG13_CGATGT_L002.bam FG14_TGACCA_L002.bam
FG15_ACAGTG_L002.bam FG16_GCCAAT_L002.bam FG17_CAGATC_L002.bam
FG18_CTTGTA_L002.bam FG1_CGATGT_L001.bam FG21_AGTCAA_L002.bam
FG22_AGTTCC_L002.bam FG23_ATGTCA_L002.bam FG24_CCGTCC_L002.bam
FG25_GTCCGC_L002.bam FG26_GTGAAA_L002.bam FG2_TGACCA_L001.bam
FG3_ACAGTG_L001.bam FG4_GCCAAT_L001.bam FG5_CAGATC_L001.bam
FG6_CTTGTA_L001.bam FG7_AGTCAA_L001.bam FG8_AGTTCC_L001.bam
FG9_ATGTCA_L001.bam Hygromycin_B_x0_5-1_S46.bam Hygromycin_B_x4-
1_S55.bam Hygromycin_B_x4-2_S56.bam Lot2_No-drug-2_S71.bam
Miltefosine_x0_5-1_S16.bam Miltefosine_x0_5-3_S18.bam
Miltefosine_x2-1_S22.bam Simvastatin_x4-2_S95.bam Terbinafine_x1-
1_S76.bam Terbinafine_x4-2_S83.bam U73122_x0_5-1_S31.bam U73122_x2-
3_S39.bam U73122_x4-1_S40.bam

conda deactivate
```

#STEP 8.3 - Sort the data in Jupyter notebooks in Python

see: A1163_UNION_TPM_GH.pdf

- Calculates the TPM
- Does expression cut-off
- Makes new GTF of all LncRNA and potential novel protein coding in the A1163 that reach expression level required
- Also make Hierarchical clustering of log2 transformed normalized TPM values (fold change relative to mean expression level for each gene)
- Density plot of $\log_2$(TPM) per loci after expression level filter, median values shown.
- Performs MannWhitneyU tests on the data

---

9 Further processing of data: GC-content, Length distribution, Genome coverage, hit clustering, exon number per transcript

---

#STEP 9.1 – First make some fasta files needed for processing using GTFs created in STEP 8.3 **A1163_UNION_TPM_GH.pdf**

Other bash commands are written in the PDF files where needed.

```
# Get transcripts in fasta

conda activate GFF_utils

gffread -w A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.fasta -W -F -g
Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.gtf

fasta_formatter -i A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.fasta -o
formatted_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.fasta

gffread -w A1163PT_UX_6_8_TPM_CUTOFF_6_8.fasta -W -F -g
Aspergillus_fumigatusa1163.ASM15014v1.dna.toplevel.fa
A1163PT_UX_6_8_TPM_CUTOFF_6_8.gtf

fasta_formatter -i A1163PT_UX_6_8_TPM_CUTOFF_6_8.fasta -o
formatted_A1163PT_UX_6_8_TPM_CUTOFF_6_8.fasta

conda deactivate
```

 # Attached PDF files from Jupyter Notebooks with detailed protocols:

- Density plots of GC-content of transcripts and lengths of transcripts
  - **A1163_UNION_after_TPM_GC-CONTENT_Lengths_GH.pdf**
- Coverage of genome by lncRNA genes and Novel protein-coding genes, and how much they overlap themselves
  - **A1163_genome_coverage_GH.pdf**
- Number of exons per transcript
  - **A1163_UNION_exon_counting.pdf**
- Distribution of hits along chromosome

    - **A1163_LNCRNA_cluster_genome_50000_GH.pdf**
- Genome GC-Content
    - **A1163_GC_content_features_GH.pdf**