How to use the OR pipeline

FRIDAY, 7/22/2022

This text is meant to be a guide for anyone who needs to use the OR pipeline that Dr. Yohe and I have made. This will detail the actual code, the file structure, the usage, and any debugging required to use the pipeline. This will allow a new lab member to take over my current Human_OR, Fossil_OR, and Bat_MC1R projects. It will also allow someone to set up the programs necessary for their own similar project.

Setting up the scripts and file structure

image.png

Here is a picture of the linux cluster terminal showing the proper set up for the pipeline. The project should be given its own folder (in this example it is called Human_OR_Final). This project folder will need to contain three subfolders named "blast", 'blastdb", and "components". The rest of the files seen in the folder are the genomes (in this case human genomes) which the pipeline will run on.

The "blast" folder is where the output from the blast program will go. The "blastdb" folder will contain the databases that the blast program will create for each genome. The "components" folder will contain all the scripts and the files needed to run the scripts.

The "components" folder will need to contain the following files:
OR_Final.sh
OR_Clean_Final.sh
gunzip.sh
files.txt
geneious_parser.py

Copying and pasting the code from each of these files into an empty file with the correct name will allow you to set up the pipeline.

OR_Final.sh is a script that controls the running of hte blast program. In this example it contians the following code:

```
#!/bin/bash
#SBATCH --partition=Orion
#SBATCH --time=96:00:00
#SBATCH --mem=124GB
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=8
#SBATCH --array=1-25

module load blast

file=$(sed -n -e "${SLURM_ARRAY_TASK_ID}p" /projects/yohe_lab/Human_OR_Final/components/files.txt)
```

```
cd /projects/yohe_lab/Human_OR_Final/

makeblastdb  -in /projects/yohe_lab/Human_OR_Final/"$file" -title "${file%.*}" -dbtype nucl -out /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}"
blastn -query /projects/yohe_lab/Human_OR_Final/components/blast_reptileChemo_TAAR.fasta -db /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}" -num_threads 8 -num_alignments 5 -outfmt 6 -out /projects/yohe_lab/Human_OR_Final/blast/"${file%.*}"_chemo.blastn
makeblastdb  -in /projects/yohe_lab/Human_OR_Final/"$file" -title "${file%.*}" -dbtype nucl -out /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}"
tblastn -query /projects/yohe_lab/Human_OR_Final/components/blast_reptileChemo.fasta -db /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}" -num_threads 8 -num_alignments 5 -outfmt 6 -out /projects/yohe_lab/Human_OR_Final/blast/"${file%.*}"_chemo.tblastn
```

*The way in which this code functions will be explianed in a later section.

OR_Clean_Final.sh is a script that cleans up the initial blast output into a more accessible form. In this example it contains the following code:

```
#!/bin/bash
#SBATCH --partition=Orion
#SBATCH --time=72:00:00
#SBATCH --mem=64GB
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=8

#post-processing script for OR_Final.sh data data
cd /projects/yohe_lab/Human_OR_Final/blast
#merge TAAR and OR data
for file in *_chemo.tblastn
do
	cat "$file" "${file%.*}".blastn > "$file"merged.chemo.blastn
done

module load bedtools2
module load blast

cd /projects/yohe_lab/Human_OR_Final/blast

for file in *_chemo.blastn
do
	python /projects/yohe_lab/genomeGTFtools-master/blast2gff.py -b "$file" -F -l 100 -s 0.2 > "${file%.*}".output.gff3
done

for file in *.gff3
do
	awk -vOFS='\t' '{$3 = $9; print}' "$file" > "$file".name
done
```

```
cd /projects/yohe_lab/Human_OR_Final/blast
for file in *.output.gff3.name
do
	bedtools getfasta -fi /projects/yohe_lab/Human_OR_Final/"${file%_*.*.*.*}".fna -bed
/projects/yohe_lab/Human_OR_Final/blast/"$file" -s -name -fo
/projects/yohe_lab/Human_OR_Final/blast/"$file".fa.out
done

#remove exact duplicates
for file in *.out
do
	~/bbmap/dedupe.sh in="$file" out="${file%.*.*}".fasta ow=t
done

#move to your home directory
cd ~/ora-1.9.1/scripts
for file in /projects/yohe_lab/Human_OR_Final/blast/*.name.fasta
do
	perl or.pl -sequence "$file" -a -d --sub "$file".sub.fasta > "$file".ORs.fasta
done

#remove empty first line from fasta file to make up for ORA bug
#this can be run repeatedly without any dataloss
#I added this script to the genomeGTFtools folder for organization

cd /projects/yohe_lab/Human_OR_Final/blast
python3 /projects/yohe_lab/genomeGTFtools-master/ORA_Fix.py

#re-remove duplicates
for file in *.fixed.fasta
do
	~/bbmap/dedupe.sh in="$file" out="${file%.*.*.*}".ORs.clean.fasta ow=t
done
```

gunzip.sh is a utility script that is used to unzip the downloaded genomes. In this example it contains the following code:

```
#!/bin/bash
#SBATCH --partition=Orion
#SBATCH --time=72:00:00
#SBATCH --mem=124GB
#SBATCH --nodes=12
#SBATCH --ntasks-per-node=12

cd /projects/yohe_lab/Human_OR_Final

gunzip *.gz
```

files .txt is a file that contains a list of the file names of th e genomes from the main folder. In this example it contains the following text:

```
GCA_000002115.2_WGSA_genomic.fna
GCA_000212995.1_HuRefPrime_genomic.fna
GCA_000252825.1_HuRef_diploid_genomic.fna
GCA_000365445.1_CSA_genomic.fna
GCA_001292825.2_HS1011_v1.1_genomic.fna
GCA_001524155.4_NA19240_prelim_3.0_genomic.fna
GCA_001712695.1_KOREF1.0_genomic.fna
GCA_002077035.3_NA12878_prelim_3.0_genomic.fna
GCA_002180035.3_HG00514_prelim_3.0_genomic.fna
GCA_003634875.1_HG00733_Phased_Diploid_genomic.fna
GCA_011064465.2_Ash1_v2.2_genomic.fna
GCA_014905855.1_ASM1490585v1_genomic.fna
GCA_015476435.1_HC1-N_genomic.fna
GCA_016695395.2_mHomSap3.mat_genomic.fna
GCA_016700455.2_mHomSap3.pat_genomic.fna
GCA_018873775.2_hg01243.v3.0_genomic.fna
GCA_020497085.1_KOREF_S1v2.1_genomic.fna
GCA_020497115.1_PGP1v1_genomic.fna
GCA_021950905.1_HG002.pat.cur.20211005_genomic.fna
GCA_021951015.1_HG002.mat.cur.20211005_genomic.fna
GCA_022833125.2_ASM2283312v2_genomic.fna
GCF_000001405.40_GRCh38.p14_genomic.fna
GCF_000002125.1_HuRef_genomic.fna
GCF_000306695.2_CHM1_1.1_genomic.fna
GCF_009914755.1_T2T-CHM13v2.0_genomic.fna
```

This is required for the OR_Final.sh script to run in an reasonalbe amount of time. This file can be created by going to the main folder an typing

```
ls > files.txt
```

The resulting file.txt file will need to be opened in a text editor in order to trim its contents such that only the names of genome files remain. It will then need to be moved to the "components" folder.

**Geneious_parser.py is too large to be displayed here, but it can be found in the same folder as this guide is in benchling.*

How the scripts work

OR_Final.sh

The function of the OR_Final.sh script is described below.

```
#!/bin/bash
#SBATCH --partition=Orion
#SBATCH --time=96:00:00
#SBATCH --mem=124GB
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=8
#SBATCH --array=1-25
```

This first section is required to make the script run on hte supercomputer cluster. The is section must be commented out with "#" to work. The first line indicated that this is a bin/bash file. The second line idicates which part of the cluster the proram will run on. In this case it will run on the "Orion" section, which is used fro general computation. The third line indicates how long the script will be allowed to run before it stops itself. Always make this time longer than you actually need so that the script doesnt ever stop when its not done. The fourth line idicates how much memory will be allocated to the running of the program, in this case it will have 124GB of ram to work with. The fifth line indicates how many "nodes" (computer processors) will be allocated to the program. The sixth line idicates how many nodes will be given to each task, even though it says "ntasks-per-node" it actually means "nodes per task". The seventh line is the most important when running blast on multiple genomes. This line allows us to run several instances of blast in parralel instead of sequentially. This line must be changed so that the last number (in this case 25) is the same as the number of genomes that you would like to run in blast.

`module load blast`
This line loads the blast program so that it can be used. Blast will not work without this.

`file=$(sed -n -e "${SLURM_ARRAY_TASK_ID}p" /projects/yohe_lab/Human_OR_Final/components/files.txt)`
This line is also extremely important. It tells the cluster which genomes blast will need to be run on, and it is this line that allows them to run in parralel. As you can see, this line references the "files.txt" file in the components folder, and this is how it figures out which genomes to run on.

`cd /projects/yohe_lab/Human_OR_Final/`
This line is inteneded to make certain that the script is in the correct folder when it starts to run blast. The end of the file filepath (in this case "/Human_OR_Final/") should be changed to the name of your project's folder.

`makeblastdb  -in /projects/yohe_lab/Human_OR_Final/"$file" -title "${file%.*}" -dbtype nucl -out /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}"`
This line creates a blast database in preperation for the actuall blast search. The -in parameter tells blast which genome to make a database for. The -title parameter tells blast what to call the newly created database. The -dbtype parameter tells blast what type of data is in the file it is making adatabae out of. In this case it is making adatabase out of a nucleotide genome so we use the "nucl" parameter to idiate nucleotide. The -out parameter tells blast where to the newly created genome, in this case it goes in the blastdb folder.

`blastn -query /projects/yohe_lab/Human_OR_Final/components/blast_reptileChemo_TAAR.fasta -db /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}" -num_threads 8 -num_alignments 5 -outfmt 6 -out /projects/yohe_lab/Human_OR_Final/blast/"${file%.*}"_chemo.blastn`
This is the line that actually runs blast. The first part, "blastn" indicates which type of blast we are running. In this case, we are searching a nucleotide database using a nucleotide query, so we use blastn. The -query parameter tells blast what data we are searching for in the genome. In this case, it is a collection of trans-amine accosted receptors (TAARs) in a file called "blast_reptileChemo_TAAR.fasta. The -db parameter tells blast which database to search in. The line of parameters that says " -num_threads 8 -num_alignments 5 -outfmt 6" is used to put the blast output in the proper format so that it works with programs later in the pipeline. The -out parameter tells blast where to put the blast output, in this case, it goes into our "blast" folder.

`makeblastdb  -in /projects/yohe_lab/Human_OR_Final/"$file" -title "${file%.*}" -dbtype nucl -out /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}"`
`tblastn -query /projects/yohe_lab/Human_OR_Final/components/blast_reptileChemo.fasta -db /projects/yohe_lab/Human_OR_Final/blastdb/"${file%.*}" -num_threads 8 -num_alignments 5 -outfmt 6 -out /projects/yohe_lab/Human_OR_Final/blast/"${file%.*}"_chemo.tblastn`

This second set of lines does essentially the same thing as the first two but uses a different type of blast and a different query. this is done in my example because we have a list of TAAR genes in nucleotide format, and we have a list of all the other OR genes in protein format. The fact that we have two queries necessitates running two different versions of blast. This may not be the case for you, depending on the format of your query file and where you got it from. This creates the problem of having two sets of output files for each genome going into the next step, which will need to be addressed, but again this may not be the case for you.

OR_Clean_Final.sh

```
#!/bin/bash
#SBATCH --partition=Orion
#SBATCH --time=72:00:00
#SBATCH --mem=64GB
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=8
```

This header is essentially that same as the one in the OR_Final.sh script, except that is does not run any of hte processes in parralel, as this script completes quickly enough to not need any optimization.

```
#post-processing script for OR_Final.sh data
cd /projects/yohe_lab/Human_OR_Final/blast
```

This line ensures that the script is in the correct folder when you run it. The end of the file filepath (in this case "/Human_OR_Final/blast") should be changed to the name of your project's folder, and the "/blast" part should be kept as is.

```
#merge TAAR and OR data
for file in *_chemo.tblastn
do
	cat "$file" "${file%.*}".blastn > "$file"merged.chemo.blastn
done
```

This part of the script merges the outputs from the blastn and tblastn runs into a sinlge file for more conveiient handleing. THis is only nessecary becasue of the two different qurey tpes I had for the human OR project. If your project only uses one type of blast, this part can be commented out or removed. Besure however that your output ends n the file extension ".blastn" otherwise you will have to change that way the output is referenced in the rest of this script.

```
module load bedtools2
module load blast
```

This section loads the modules needed to run the rest of the script.

```
cd /projects/yohe_lab/Human_OR_Final/blast
```

This is another redundant file change command to ensure we are in the right folder. The same rules apply to it as do to the one above.

```
for file in *_chemo.blastn
do
	python /projects/yohe_lab/genomeGTFtools-master/blast2gff.py -b "$file" -F -l 100 -s 0.2 >
"${file%.*}".output.gff3
done
```

This section calls a program called blast2gff.py from a folder in the yohe_lab directory. This program converts the blast output files in .gff3 files.

```
for file in *.gff3
```

```
do
        awk -vOFS='\t' '{$3 = $9; print}' "$file" > "$file".name
done
```
This section trims the .gff3 files so that they only contain the required information for the next step. It also give the new output files the ".name" extension.

```
cd /projects/yohe_lab/Human_OR_Final/blast
```
Another redundant file change line.

```
for file in *.output.gff3.name
do
        bedtools getfasta -fi /projects/yohe_lab/Human_OR_Final/"${file%_*.*.*.*}".fna -bed
/projects/yohe_lab/Human_OR_Final/blast/"$file" -s -name -fo
/projects/yohe_lab/Human_OR_Final/blast/"$file".fa.out
done
```
This section uses the bedtools funtion "getfasta" to find the sequences in the original genomes that the coordinates in the .gff3.name files reference. The resulting output files contain these sequences and end in the file extension ".fa.out".

```
#remove exact duplicates
for file in *.out
do
        ~/bbmap/dedupe.sh in="$file" out="${file%.*.*}".fasta ow=t
done
```
This section uses the dedupe.sh program from the bbmpa package to remove any sequences that are exact duplicates of each other. In my case the bbmap package is installed in my home directory, but you will have to install it in yours. Talk to Dr. Yohe for information on how to do this.

```
#move to your home directory
cd ~/ora-1.9.1/scripts
for file in /projects/yohe_lab/Human_OR_Final/blast/*.name.fasta
do
        perl or.pl -sequence "$file" -a -d --sub "$file".sub.fasta > "$file".ORs.fasta
done
```
This section calls the olfactory-receptor-assigner (ORA) program and uses it to characterize the genes you have found with blast. ORA will determine is the genes is really an olfactory receptor, if it is a functional olfactory receptor, and what class of olfactory receptor it falls into. Ask Dr. Yohe to explain how this works if you want more details. The ORA package will also need to be installed in your home directory, and Dr. Yohe will need to help you with this aswell because I don't quite remember how she got me to do it. The files output by ORA will be in fasta format with headers describing the genes followed by their nucleotide sequences. The output files will have the extension ".ORs.fasta".

```
#remove empty first line from fasta file to make up for ORA bug
#this can be run repeatedly without any dataloss
#I added this script to the genomeGTFtools folder for organization
```

```
cd /projects/yohe_lab/Human_OR_Final/blast
python3 /projects/yohe_lab/genomeGTFtools-master/ORA_Fix.py
```

This is a program I made to fix a bug in the ORA program. The ora program leaves a blank line at the top of all its output files, which throw off alot of other programs that expect a fasta file to start with a ">" header. This program fixes this. I have placed the program in the genomeGFTtools-master folder in the yohe_lab folder so it can be easily called when needed.

```
#re-remove duplicates
for file in *.fixed.fasta
do
	~/bbmap/dedupe.sh in="$file" out="${file%.*.*.*}".ORs.clean.fasta ow=t
done
```

This section re-removes any exact duplicates as ORA shifts the reading from of some of the genes. The resulting output files and the final output of this script, they have the file extension ".ORs.clean.fasta".

How to use the pipeline

Once the file structure and programs are set up as described in the first section, and the genomes have been uploaded to the cluster into the appropriate folder as described, then the pipeline can be run.

Step 1: run the OR_Final.sh script. This is done by navigating to the components folder and running the following command:

```
sbatch OR_Final.sh
```

This script may take a day or more to complete depending on the size and number of genomes you are running. To see the progress of the script use the command:

```
squeue -u [your username]
```

This will show you a list of all the instances of blast that are running, nad how long each has been running.

Step 2: run the OR_Clean_Final.sh script using the following command:

```
sbatch OR_Clean_Final.sh
```

Again the progress of this process can be tracking using:

```
squeue -u [your username]
```

This script should take a few hours to run.

Step 3: Move into the "blast" folder and create a new directory to separate out the final data in. I named mine "Human_ORs", name yours based off of your project. use the command:

```
cp *.ORs.clean.fasta [Name of Your Folder]/
```

to move all of the final files into this new folder. At this point, this folder can either be downloaded using sftp onto your desktop, or it can be moved into another folder on the cluster. But it must be moved out of the blast folder for the next step to work properly. In my case, I made a new directory called "analysis" in my main "Human_OR_Final" project folder and I moved the final data folder there.

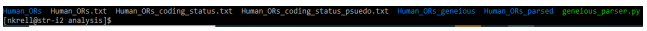
Step 4: Move the "geneious_parser.py" program into the same folder as the final data folder. This program will only work if it is "next to" the final data folder. It will not work if it is *in* the final data folder. Run the command:

```
python3 geneious_parser.py
```

The program will prompt you for hte name of the folder containing the data to be analysed: type in the folder name exactly (in my case it was "Human_ORs").

The geneious parser program will take up to an hour to run. It will output updates to the command line as it completes its parsing of each genome. The final results will look like this:

image.png



There will be 3 text files, each containing a csv (comma separated value) file showing the totlas of each type of gene that was detected by ORA. First file (in this example "Human_ORs.txt") will have the overal cou8nt of each gene family. The second one (in this example "Human_ORs_coding_status.txt") will have the totlas of only the genes that were intact. The final file (in this case "Human_ORs_coding_status_psuedo.txt") will have totlas of only the gens that were psudogenized. These cvs file can be loaded inot excle or R in order to make graphs. A folder whose name ends with "_geneious" will also be created. This folder will have the genes and their sequences sorted by family in a file structure that can be imported directly into the geneious program for alignment work.