PLASANN (Plasmid Annotation Pipeline)

LEGEND:

SET-UP YOUR COMPUTER:

- REQUIRED PACKAGES:
  (The required packages can be found in the corresponding folder containing all files)
  - Conda
  - Biopython
    - `conda install -c conda-forge biopython`
  - Prokka
    - `conda env create --file ./scripts/prokka_env.yml`
      - prokka_env.yml is in the scripts folder
  - Abricate
    - `conda env create --file ./scripts/abricate_env.yml`
      - abricate_env.yml is in the scripts folder
  - CD-HIT and PSI-CD-HIT
    - File containing programs for CD-HIT is already stored in the Jan. 16th folder

- COMPUTER SETTINGS:
  - The program does take some time to run due to how much data needs to be processed for each plasmid. To make sure that there are no issues running the code in its entirety, set up your computer so it doesn't turn off due to inactivity. To do this please go into your computer's settings and set up your computer so it doesn't turn off your screen or turn on the screen saver when inactive.
    - Otherwise, your computer will enter its inactive state and shut down the program, leading to connection errors

SCRIPT INFORMATION:

- **ORDER:**
  PlasAnn.py

---bgbm.sh
------annotate.py
------prokka_nr.py
---------prokka_def.sh
------------prokka2go.py
---------prokka.sh
------------prokka2go.py
------prokka_process.py
------abricate_dw.py
------abricate_test.py
------gbk_update.py
------visual_map.py
Lr_file.py
linear-regression.py

- **SCRIPT LEGEND:**
    - Organization Scripts (call on pipelines separately)
    - Annotation Pipeline
    - Visual Pipeline
    - Linear Regression Pipeline

- **SCRIPTS:**

PlasAnn.py

- *INPUT:* See description below
    - When you initially run PlasAnn, it will ask you to specify whether to run the entire script or just the linear regression pipeline. If you want to run the entirety of the program, you may specify what plasmids you want to rerun the program on (including ALL). Follow the instructions as they appear.
- *OUTPUT:* TSV final annotation, plasmid map, and acquisition cost linear regression analysis for all plasmids.
- *DESCRIPTION:*
    - PlasAnn is the script a user must call to start annotating plasmids. It reads through my fasta folder, which contains a fasta file for each plasmid I want analyzed, one fasta file at a time, taking the plasmid name from the fasta file and passing it on to bgbm.sh, a shell script calling each script in the pipeline in their respective order in order to annotate the given plasmid.
    - You will be prompted to enter certain inputs depending on whether you want to run the entire pipeline or just the linear regression script.

bgbm.sh

- *INPUT:*
  - Plasmid name
- *OUTPUT:*
  - TSV final annotation
  - GFF final annotation
  - GBK final annotation
  - plasmid map
- *DESCRIPTION:* bgbm.sh is a shell script that calls each subsequent script in the annotation and visualization pipelines in their respective order in order to annotate and visualize the given plasmid. It utilizes conda to help isolate each script that needs its own environment in order to run properly.
  - Order in which these scripts are called:
    - annotate.py
    - prokka_nr.py
    - prokka_process.py
    - abricate_dw.py
    - abricate_test.py
    - gbk_update.py
    - visual_map.py

annotate.py

- *INPUT:* Plasmid name
- *OUTPUT:*
  - Plasmids folder, which will contain a folder for each plasmid
    - gb_match folder in the plasmid folder for the plasmid being processed
      - Ie. ./plasmids/plasmid_name/gb_match
  - matches.csv=csv file that contains the accession ID's of the initial BLAST gbk files that match our standards for including them in our final annotation of the given plasmid. Located in plasmids folder
  - Gbk files for each accession ID that was an initial match to our standards for including them in our annotation. Located in
- *DESCRIPTION:* Through a BLAST Search and input fasta file, program finds matching accession ID's to given sequence and tests their reliability. Passing accession ID's are compiled into a CSV file. (1) Run BLAST (2) Read in BLAST file and parse through each accession ID, downloading their gbk files and determining if the given accession ID is fit for use (3) Compile good accession ID's into CSV file

prokka_nr.py

- *INPUT*:
    - Plasmid name
    - matches.csv for given plasmid
- *OUTPUT*:
    - Prokka output for each accession ID
    - prokka2go.txt file containing the corresponding GO ID and functional label for each identified gene
    - Plasmid_name-go.tsv
    - New matches.csv file containing a more limited list of accession ID matches
- *DESCRIPTION:* (1) Creates directories for storing prokka results for each accession ID for the given plasmid (2) Runs prokka in default mode (no gbk file uploaded) by calling prokka_def.sh shell script and runs prokka with the protein flag once for each accession ID in matches.csv, utilizing each accession ID's corresponding gbk file, by calling the prokka.sh shell script. These shell scripts call on prokka2go.py to process the given gene annotations and add their GO identified functions to the initial annotation of the plasmid (4) Create plasmid_name-go.tsv for each accession ID, which merges the prokka generated annotation of the plasmid and the matching GO ID's into one tsv file (5) Create a new matches.csv file to compile most accurate gbk matches based on the prokka output

prokka_def.sh

- INPUT:
    - Output directory for prokka
    - Plasmid name
- OUTPUT:
    - Prokka generated files in pk_results folder in the default folder in gb_match
    - prokka2go.txt file generated by prokka2go in the corresponding default folder in gb_match that contains the GO-ID and corresponding functional label for each gene identified by the default prokka script
- DESCRIPTION: Utilizing conda, this shell script calls on the prokka_env to run prokka without providing a gbk file (ie. default mode). Once prokka is done running, it deactivates the prokka_env environment and calls on prokka2go in order to process the functional labels for each identified gene.

prokka.sh

- *INPUT:*
    - Output directory for prokka
    - Plasmid name
    - Directory for corresponding accession ID gbk file folder
    - Accession ID being processed
- *OUTPUT:*
    - Prokka generated files in pk_results folder in the corresponding folder for the accession ID being looked at
    - prokka2go.txt file generated by prokka2go in the corresponding accession ID folder in gb_match that contains the GO-ID and corresponding functional label for each gene identified by prokka
- *DESCRIPTION:* Utilizing conda, this shell script calls on the prokka_env to run prokka with the --proteins flag and the given accession ID gbk file. Once prokka is done running, it deactivates the prokka_env environment and calls on prokka2go in order to process the functional labels for each identified gene.

prokka2go.py

- *INPUT*:
    - Input location of gbk file created by prokka
    - Output file location
    - Plasmid name
- *OUTPUT*:
    - prokka2go.txt
- *DESCRIPTION*: Code is based upon prokka2kegg.py, which was built by Heyu Lin. Currently, prokka2go parses through the given gbk file, looking at one gene at a time. If a gene is identified, the script looks to see if it has a cited UNIPROT ID. If not, it will do a query search for that gene in UNIPROT to get a list of matching UNIPROT ID's, pulling out the first reviewed UNIPROT match or first unreviewed match if there are no reviewed matches, saving the results for each gene onto a list. This list is then passed onto get_go_id, which accesses the given UNIPROT entry for that gene and parses through it to obtain the listed GO ID's, prioritizing the GO ID's labeled as being biological. If no GO ID's are listed, the keywords listed under that UNIPROT entry are looked at for labeling. The matching GO ID for each gene is concatenated onto a list, which is passed onto the output function that writes the results to prokka2go.txt, which is located in the folder for the accession file that was just processed by prokka.

prokka_process.py

- *INPUT*:

    - Plasmid name
- *OUTPUT*:
    - mergedgo.tsv, which contains every annotation created by prokka + prokka2go for each accession ID
    - final.tsv, which contains the consensus final annotation for the given plasmid
- *DESCRIPTION*: Create a mergedgo.tsv file that contains every annotation generated by prokka + prokka2go, sort through this merged annotation and create a consensus annotation for the given plasmid.

abricate_dw.py

- *INPUT*:
    - Plasmid name
- *OUTPUT*:
    - ab_results folder for each accession ID listed in matches.csv for that particular plasmid
    - 2 text files per database (total of 8)
        - NCBI, CARD, and RESFINDER list matching antibiotic resistance genes
        - PLASMIDFINDER lists the matching incompatibility group for the given plasmid
- *DESCRIPTION*: Run abricate w/ncbi, card, resfinder, and plasmidfinder databases for each accession ID match

abricate_test.py

- *INPUT*:
    - Plasmid name
- *OUTPUT*:
    - Edited final.tsv file that contains the final annotation of the given plasmid
    - final.gff file that contains the final annotation of the given plasmid + incompatibility group + fasta sequence
- *DESCRIPTION*: After running abricate with 3 different antibiotic resistance databases for each accession ID, check that 2/3 databases found the same resistance genes, and write the names of these confirmed consensus resistance genes to the final annotation for the plasmid. Also add the incompatibility group to the final gff for the plasmid found by plasmidfinder

gbk_update.py

- *INPUT*:

    - Plasmid name
- *OUTPUT*:
    - Edited final.gbk file that contains the final annotation of the given plasmid + product information + translation information
- *DESCRIPTION*: Once the finalized annotation is made, take the gbk file generated by prokka in its default setting and make a copy of it. In this copy, called final.gbk, update gene names in the corresponding locus locations and any other information pertaining to those genes.

visual_map.py

- *INPUT*:
    - Plasmid name
- *OUTPUT*:
    - PDF of plasmid map for given plasmid
- *DESCRIPTION*: Using GenomeDiagram, visual_map.py parses through the final annotation for the given plasmid to create a plasmid map of the specified plasmid. Each gene on the plasmid is labeled to fit one of the following categories of interest: Conjugation, Antibiotic Resistance, Stress-Response, Toxin, Metabolism, Other, Unknown (More will be added, such as Replication, Background/Backbone, and Antitoxin)

lr_file.py

- *INPUT:*
    - File containing acquisition cost information for all or some of the plasmids analysed previously in the script
- *OUTPUT:*
    - Linear Regression file, which will be used in the python-script to run a linear regression analysis on the plasmids and the acquisition cost data, with the following information:
        - Binary plasmid gene presence information for each plasmid
        - Respective number of gene functions for each function
        - Acquisition cost data
        - Plasmid data (ie. source, etc.)
- *DESCRIPTION:*
    - Using PSI CD-Hit, I create a list of unique homologous genes found across all of the plasmids analysed for the purposes of adding binary information about gene presence for each plasmid in the final linear regression file. I append acquisition cost data, plasmid data, gene presence, and gene function data for each plasmid to this file so the file

    - contains a table containing data for each plasmid in each corresponding row.

linear-regression.py

- *INPUT:*
    - PlasmidRegression.csv created by lr_file.py
- *OUTPUT:*
    - Print linear regression results
    - May make file containing data in case results need to be used by another program
- *DESCRIPTION:*
    - Print linear regression results based on gene presence data, plasmid incompatibility group, gene groups, and acquisition cost data.

<u>**RUNNING PLASANN:**</u>

1. Download the folder uploaded in the Jan. 16th folder onto your computer.
    1. It contains all scripts involved in the program + folders needed at this current stage to run it
2. Install conda + abricate + prokka (see SET-UP YOUR COMPUTER)
3. On your Mac Terminal, use the cd command to access the "PlasAnn" folder (you can right click on the folder and click "Get Info" to see the file path for the folder to access it)
4. Run `python PlasAnn.py`:
    1. Once you run it, the code will prompt you to enter information to specify how to run the program.
        - i. `Enter 1`
            1. For running ONLY the linear regression pipeline. Do this ONLY if you have all the annotation data OR you ran the FULL plasmid annotation pipeline.
        - ii. `Enter 0`
            1. Run the FULL pipeline (annotation + linear regression).
            2. **WARNING:** Depending on how many plasmids you are analyzing, running the full pipeline will take a while. If you decide to do this, please go into your computer's settings and <u>set up your computer so it doesn't turn off your screen or turn on the screen saver when inactive</u>. Do not turn off your computer while the program runs; this could lead to errors or could cause the program to stall!
            3. The program will ask you which plasmids you would like to rerun through the pipeline (if any or if all).

5. Wait for the program to finish! Do check on it from time to time to make sure that it's running and let me know if any issues/questions arise.

ERROR HANDLING

- Prokka/Abricate
    - If any errors occur with Prokka or Abricate (ie. tbl2asn not running OR prokka is not outputting any files) terminate the program. You will now need to uninstall prokka and/or abricate and then reinstall prokka and/or abricate. To reinstall, you will need to use the .yml files in the scripts folder. Thus, make sure that on your terminal you are in the PlasAnn directory. Below is the code you need:
        - Make sure you're in the PlasAnn folder in your terminal
            - `cd file_path/PlasAnn`
        - For uninstalling
            - `conda env remove -n prokka_env`
            - `conda env remove -n abricate_env`
        - For reinstalling
            - `conda env create --file ./scripts/prokka_env.yml`
            - `conda env create --file ./scripts/abricate_env.yml`
        - Additional Code to run:
            - `conda install -y -c biobuilds perl=5.22`
            - `conda install -y -c conda-forge parallel`
            - `conda install -y -c bioconda prodigal blast=2.2 tbl2asn prokka`
    - If the following does not work, email me ASAP (jc4919@barnard.edu). It is possible that another program may be needed to correctly install prokka or abricate.
    - If the issue persists, check that your virus blocking software or other applications are not installing or blocking prokka/abricate. Email me ASAP (jc4919@barnard.edu) as well.
- Connection Issues
    - If you are having connection issues or the error message "ConnectionResetError: [Errno 54] Connection reset by peer" appears on your terminal, you will need to terminate the program and rerun it. I can help you set up the program so you can start from where it left off so you do not have to rerun it in its entirety.
- Other
    - If at any point the program crashes or some of the scripts begin to fail, quit the program (either terminate or quit using Command-C until no scripts are running) and send me a message (jc4919@barnard.edu) about what script was failing with the error message. I will look over the issue, fix it, and send you an updated file of the failing script. I can also help you set up the program so you can start from where it left off so you do not have to rerun it.