HyAsP and ~~PlasBin~~ plASgraph

Cedric Chauve, Dept. Mathematics, SFU

Data, experiments and results: https://github.com/cchauve/ARETE_MAY_2022

Plasmid prediction problems

Input
The assembled *contigs* of a bacterial isolate, from *Illumina short-reads* sequencing data.

Problem 1: Contigs Classification (PlasGraph)
For each contig, classify it as *plasmid, chromosomal or ambiguous* (shared plasmid/chromosome sequence).

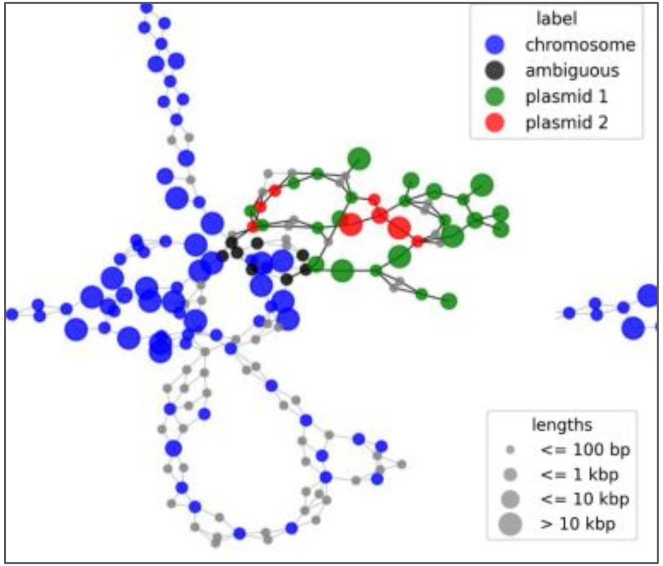
Problem 2: Contigs Binning (HyAsP)
Create groups *(bins)* of contigs, each group being expected to *originate from the same plasmid.*

Plasmid prediction problems and the assembly graph

Observations

Many assembler (e.g. Spades, Unicycler) provide an assembly graph whose nodes are contigs and edges represent possible contiguity between pairs of contigs supported by sequencing data.

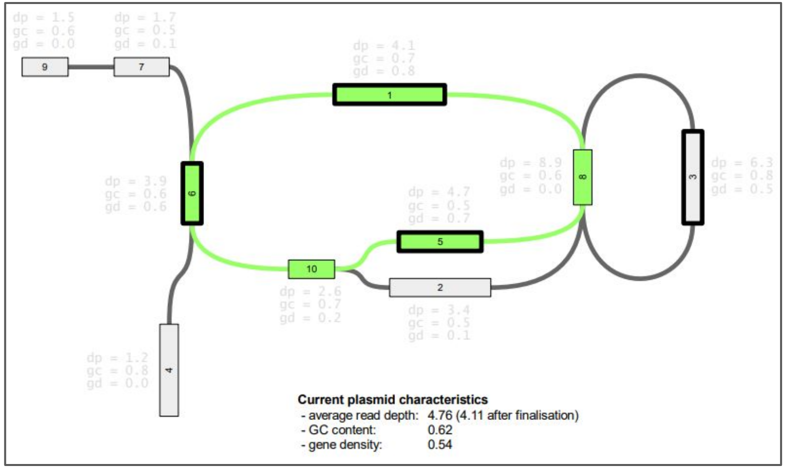
Actual plasmids in a sample are likely to correspond to groups of closely located nodes (contigs) in this graph.



Methods

Leveraging the information provided by the assembly graph to improve the accuracy of contigs classification and binning.

Contigs binning: HyAsP



HyAsP, a greedy tool for plasmids identification

Paper: https://doi.org/10.1093/bioinformatics/btz413
Code: https://github.com/cchauve/HyAsP

Algorithm: hybrid de novo and reference-based

Greedy exploration of the assembly graph to extract walks (contig bins).
Walk extension criteria:

- presence of known plasmid genes (mapping against a plasmids reference database), every walk starts from a seed, defined as a contig with a high density of plasmid genes,
- uniformity of read coverage (proxy for copy number),
- uniformity of GC content.

Installing HyAsP on cedar

HyAsP can be installed through either a Singularity container (handles all dependencies) or a python package (recommended: within a python virtual environment).

```
# Virtual environment
module load StdEnv/2020  gcc/9.3.0 blast+/2.12.0 python/3.6
virtualenv --no-download ~/hyasp_env
source ${ARETE_MAY22_HOME}/hyasp_env/bin/activate
```

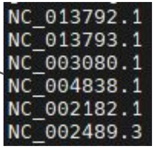
```
# HyAsP installation as a python package
git clone https://github.com/cchauve/hyasp.git
cd hyasp
python setup.py sdist
pip install dist/HyAsP-1.0.0.tar.gz
```

Running HyAsP on cedar: database creation

```
source ${ARETE_MAY22_HOME}/hyasp_env/bin/activate

# Database creation (need to be done just once)
# "A Curated, Comprehensive Database of Plasmid Sequences" 10.1128/MRA.01325-18
cd ${ARETE_MAY22_HOME}/exp/doi_10.15146_R33X2J__v2/
hyasp create doi_10.15146_R33X2J__v2_genes.fasta \
	-a ../../data/doi_10.15146_R33X2J__v2/doi_10.15146_R33X2J__v2_id.txt -d -l 500 -m 100
```

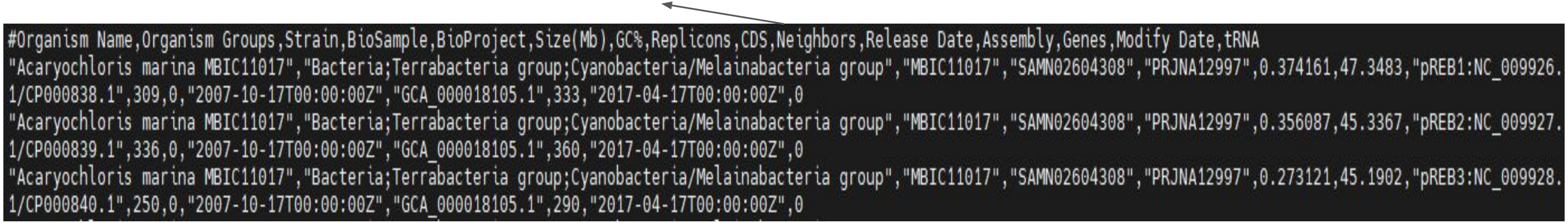
Options:
min. plasmid length 500
min. gene length 100

Created database



```
NC_013792.1
NC_013793.1
NC_003080.1
NC_004838.1
NC_002182.1
NC_002489.3
```

```
# Creating an alternate database from a list of GenBank  plasmids information
cd ${ARETE_MAY22_HOME}/ncbi_database/
hyasp create ncbi_database_genes.fasta -p plasmids.csv -d -l 500 -m 100 -t GenBank
```



```
#Organism Name,Organism Groups,Strain,BioSample,BioProject,Size(Mb),GC%,Replicons,CDS,Neighbors,Release Date,Assembly,Genes,Modify Date,tRNA
"Acaryochloris marina MBIC11017","Bacteria;Terrabacteria group;Cyanobacteria/Melainabacteria group","MBIC11017","SAMN02604308","PRJNA12997",0.374161,47.3483,"pREB1:NC_009926.
1/CP000838.1",309,0,"2007-10-17T00:00:00Z","GCA_000018105.1",333,"2017-04-17T00:00:00Z",0
"Acaryochloris marina MBIC11017","Bacteria;Terrabacteria group;Cyanobacteria/Melainabacteria group","MBIC11017","SAMN02604308","PRJNA12997",0.356087,45.3367,"pREB2:NC_009927.
1/CP000839.1",336,0,"2007-10-17T00:00:00Z","GCA_000018105.1",360,"2017-04-17T00:00:00Z",0
"Acaryochloris marina MBIC11017","Bacteria;Terrabacteria group;Cyanobacteria/Melainabacteria group","MBIC11017","SAMN02604308","PRJNA12997",0.273121,45.1902,"pREB3:NC_009928.
1/CP000840.1",250,0,"2007-10-17T00:00:00Z","GCA_000018105.1",290,"2017-04-17T00:00:00Z",0
```

Running HyAsP on cedar: processing a sample

Input: isolate assembly graph (GFA format)

Input: reference plasmid genes database

Output directory

```
source ${ARETE_MAY22_HOME}/hyasp_env/bin/activate
cd ${ARETE_MAY22_HOME}/exp/e_feacium_E7663/
INPUT=E_7663.gfa
# Mapping contigs against the reference plasmid genes database
REF1=../doi_10.15146_R33X2J__v2/doi_10.15146_R33X2J__v2_genes.fasta
hyasp map  ${REF1}  -g ${INPUT}          E7663_1_gcm.csv
hyasp filter ${REF1}  E7663_1_gcm.csv E7663_1_filtered_gcm.csv
# Compute plasmid bins
hyasp find  ${INPUT} ${REF1} E7663_filtered_1_gcm.csv ./output_1
# Important output files
output_1/contig_chains.csv                  # contig bins
output_1/putative_plasmids.fasta            # assembled sequence of putative plasmid bins
output_1/putative_plasmid_contigs.fasta # sequences of contigs in putative plasmids

# Using an alternate reference database
REF2=../ncbi_database/ncbi_database_genes.fasta
hyasp map  ${REF2}  -g ${INPUT}          E7663_2_gcm.csv
hyasp filter ${REF2}  E7663_2_gcm.csv E7663_filtered_2_gcm.csv
hyasp find  ${INPUT} ${REF2} E7663_filtered_2_gcm.csv ./output_2
```

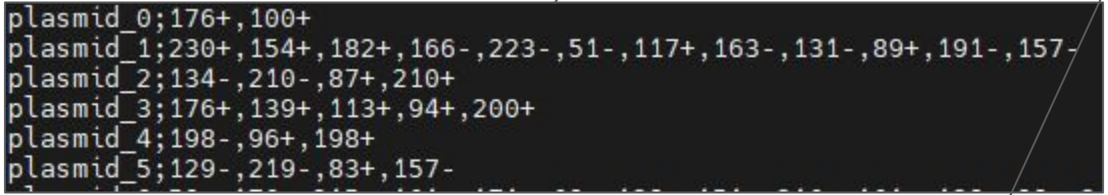
Running HyAsP on cedar: output

# Important output files

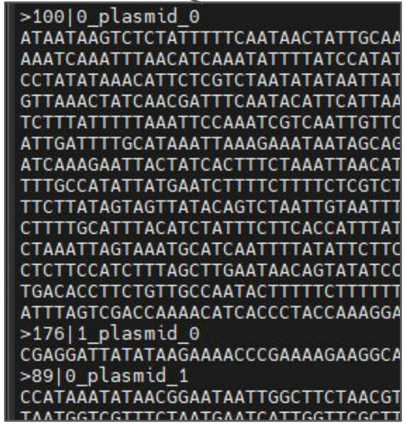
output_1/contig_chains.csv # contig bins: one ordered list of oriented contig IDs per bin

output_1/putative_plasmids.fasta # assembled sequence of putative plasmid bins

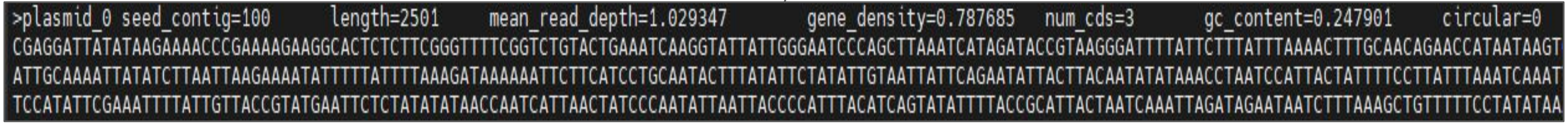
output_1/putative_plasmid_contigs.fasta # sequences of contigs in putative plasmids



```
plasmid_0;176+,100+
plasmid_1;230+,154+,182+,166-,223-,51-,117+,163-,131-,89+,191-,157-
plasmid_2;134-,210-,87+,210+
plasmid_3;176+,139+,113+,94+,200+
plasmid_4;198-,96+,198+
plasmid_5;129-,219-,83+,157-
```

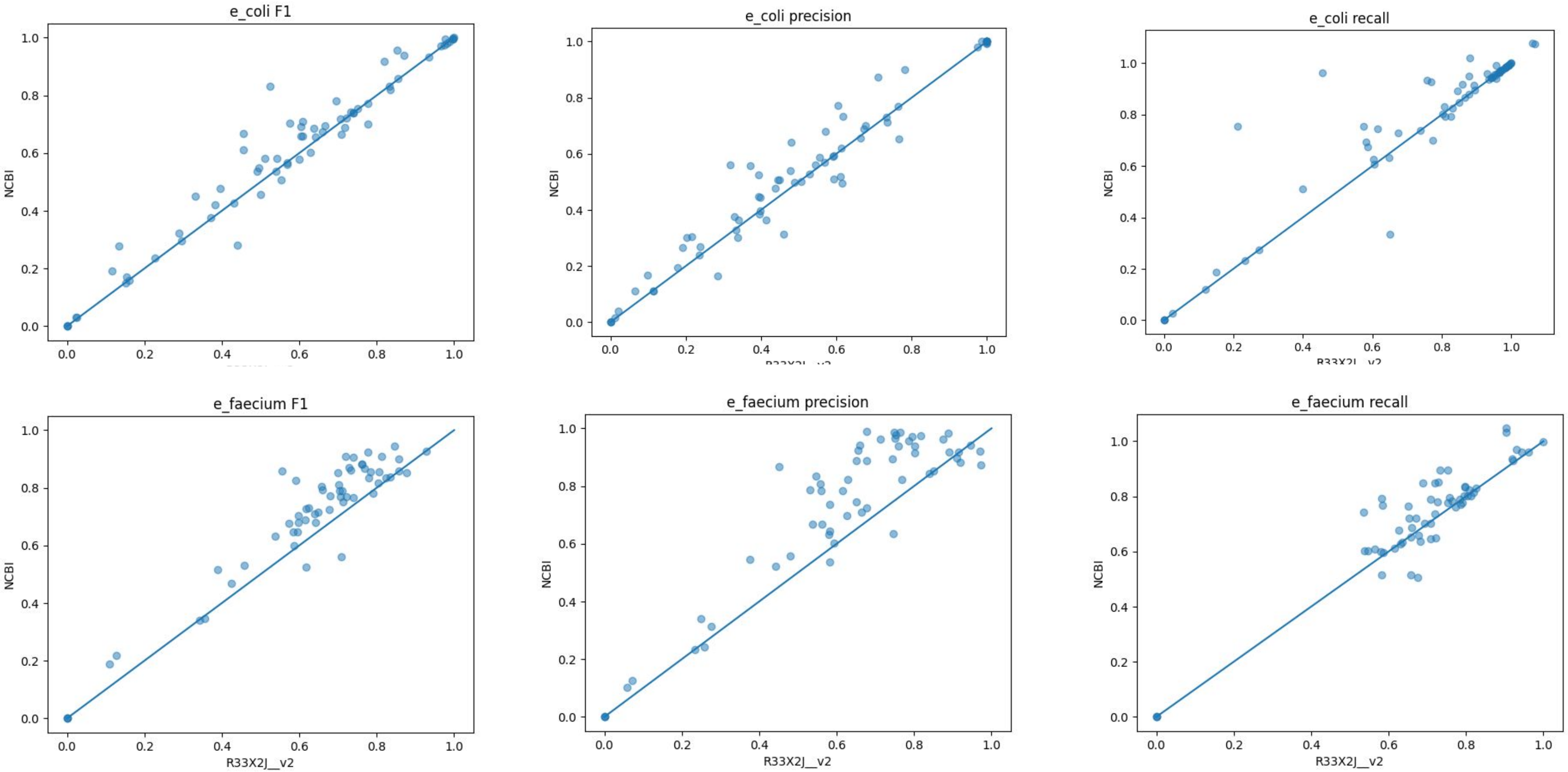


```
>100|0_plasmid_0
ATAATAAGTCTCTATTTTTCAATAACTATTGCAA
AAATCAAATTTAACATCAAATATTTTATCCATAT
CCTATATAAACATTCTCGTCTAATATATAATTAT
GTTAAACTATCAACGATTTCAATACATTCATTAA
TCTTTATTTTTAAATTCCAAATCGTCAATTGTTC
ATTGATTTTGCATAAATTAAAGAAATAATAGCAG
ATCAAAGAATTACTATCACTTTCTAAATTAACAT
TTTGCCATATTATGAATCTTTTCTTTTCTCGTCT
TTCTTATAGTAGTTATACAGTCTAATTGTAATTT
CTTTTGCATTTACATCTATTTCTTCACCATTTAT
CTAAATTAGTAAATGCATCAATTTTATATTCTTC
CTCTTCCATCTTTAGCTTGAATAACAGTATATCC
TGACACCTTCTGTTGCCAATACTTTTTCTTTTTT
ATTTAGTCGACCAAAACATCACCCTACCAAAGGA
>176|1_plasmid_0
CGAGGATTATATAAGAAAACCCGAAAAGAAGGCA
>89|0_plasmid_1
CCATAAATATAACGGAATAATTGGCTTCTAACGT
TAATGGTCGTTTCTAATGAATCATTGGTTCGCTT
```



```
>plasmid_0 seed_contig=100	length=2501	mean_read_depth=1.029347	gene_density=0.787685	num_cds=3	gc_content=0.247901	circular=0
CGAGGATTATATAAGAAAACCCGAAAAGAAGGCACTCTCTTCGGGTTTTCGGTCTGTACTGAAATCAAGGTATTATTGGGAATCCCAGCTTAAATCATAGATACCGTAAGGGATTTTATTCTTTATTTAAAACTTTGCAACAGAACCATAATAAGT
ATTGCAAAATTATATCTTAATTAAGAAAATATTTTTATTTTAAAGATAAAAAATTCTTCATCCTGCAATACTTTATATTCTATATTGTAATTATTCAGAATATTACTTACAATATATAAACCTAATCCATTACTATTTTCCTTATTTAAATCAAAT
TCCATATTCGAAATTTTATTGTTACCGTATGAATTCTCTATATATAACCAATCATTAACTATCCCAATATTAATTACCCCATTTACATCAGTATATTTTACCGCATTACTAATCAAATTAGATAGAATAATCTTTAAAGCTGTTTTTCCTATATAA
```

Running HyAsP on cedar: accuracy and database impact



Running HyAsP on cedar: accuracy and database impact

Observation 1.
HyAsP has a high recall, so is able to detect most plasmids.

Observation 2.
The precision is lower, so some putative plasmids are false positive.
It is useful to complement HyAsP by looking for plasmid-specific genes in the plasmid bins.

Observation 3.
The use of a larger, although less curated, reference database has a significant impact to increase the accuracy, especially the precision.

HyAsP: comments

Installation.

HyAsP can be installed through either a Singularity container (handles all dependencies) or a python package (recommended: within a python virtual environment).

Input.

HyAsP can take as input either an assembly graph (GFA format, e.g. from Unicycler) or the raw sequencing data, in which case it does preprocess the reads and assemble them with Unicycler9this requires to install quite a few dependencies).

Results accuracy.

- Dependent on the structure of the assembly graph (the more tangled, the less accurate).
- Performs less well on single-copy plasmids.
- Often a single plasmid corresponds to several bins (read depth often allows to join them).
- Results can be refined by searching for plasmid-specific genes in bins (a la MOB-suite).

PlasBin

PlasBin is another binning tool based on the same principle than HyAsP, but using an exact optimization method (Mixed Integer Linear Programming, MILP) in place of a greedy heuristic.

It has accuracy results slightly better than HyAsP, at the cost of a much longer computational footprint (time and memory).

It will soon been replaced by an improved MILP algorithm that uses plASgraph output and a better model of GC content deviation between chromosomes and plasmids.

Paper: https://doi.org/10.1007/978-3-031-06220-9_16
Code: https://github.com/cchauve/PlasBin

Contigs classification: plASgraph

plASgraph - using graph neural networks to detect plasmid contigs from an assembly graph

Paper: submitted

Code: https://github.com/cchauve/plASgraph

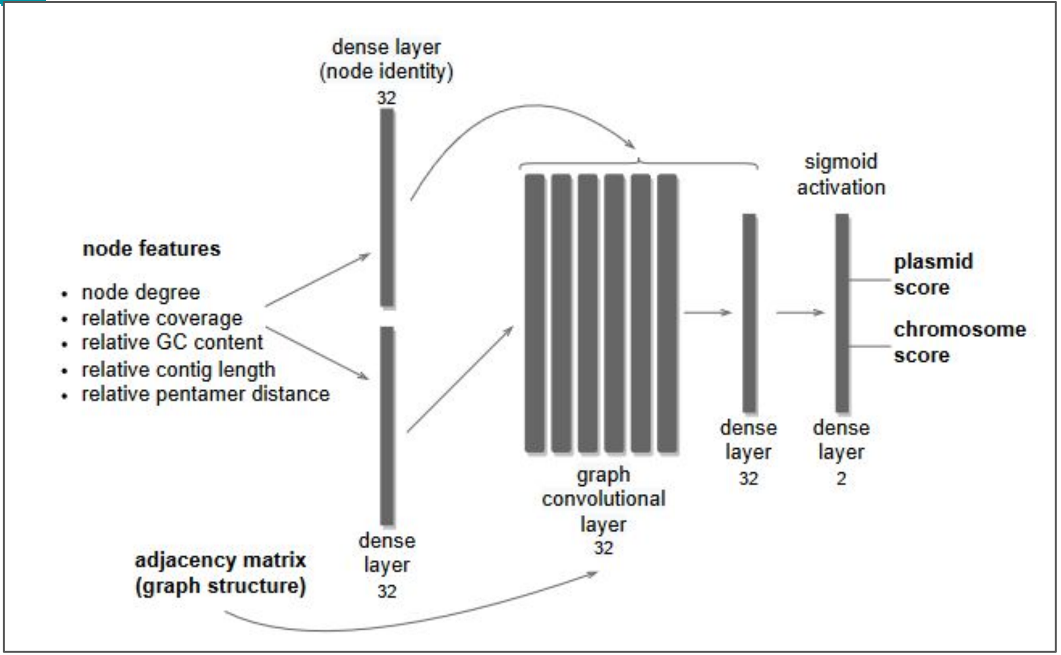
Algorithm:

Graph Neural Network

Training data: de novo

Hybrid long-reads and short-reads assemblies from *E. coli, E. faecium, K. pneumoniae*.

Hybrid (long) contigs were classified as plasmid, chromosomal, ambiguous or no_label, and short reads contigs were classified through mapping to hybrid contigs.



Installing plASgraph on cedar

```
# Virtual environment
module load python/3
python3 -m venv --system-site-packages ${ARETE_MAY22_HOME}/plasgraph_env
source ${ARETE_MAY22_HOME}/plasgraph_env/bin/activate

pip install networkx==2.6.3
pip install pandas==1.4.0
pip install numpy==1.22.2
pip install scikit-learn==0.23.1
pip install biopython==1.79
pip install matplotlib==3.5.1
pip install --no-index tensorflow==2.8
pip install spektral==1.0.8

mkdir -p ${ARETE_MAY22_HOME}/tools
cd ${ARETE_MAY22_HOME}/tools/
git clone https://github.com/cchauve/plASgraph.git
```

Running plASgraph on cedar: processing a sample

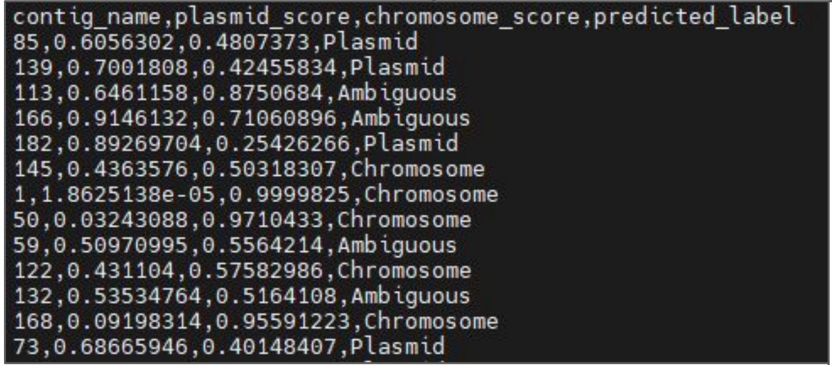
Options required to use tensorflow

```
#!/bin/bash
#SBATCH --gres=gpu:1
#SBATCH --cpus-per-task=6
#SBATCH --mem=32000M

# Virtual environment home directory
PLASGRAPH_ENV_HOME=${ARETE_MAY22_HOME}/plasgraph_env
source ${PLASGRAPH_ENV_HOME}/bin/activate

# Input
EXP_DIR=${ARETE_MAY22_HOME}/exp/e_faecium_E7663/
INPUT=${EXP_DIR}/E7663.gfa
# Running plASgraph
cd ${ARETE_MAY22_HOME}/tools/plASgraph
python plASgraph.py -i ${INPUT} -o ${EXP_DIR}/plasgraph_output/E7663_class.csv --draw_graph

# Output files
E7663_class.csv          # classification of all contigs
E7663_class_graph.png        # drawing of the assembly graph with classified contigs
```

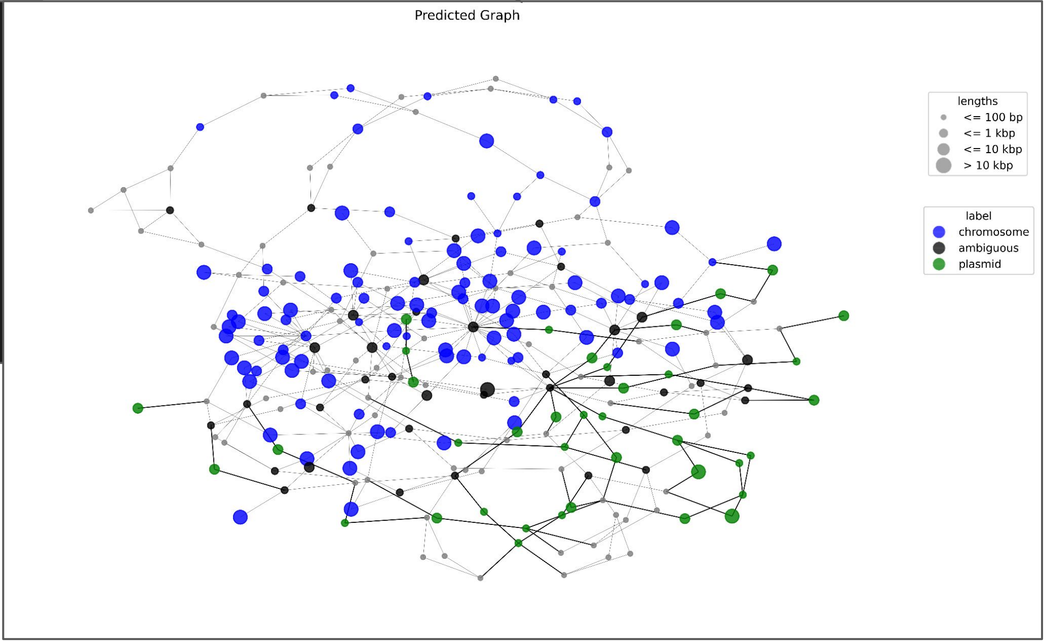
Input: assembly graph

Running plASgraph on cedar: output

# Output files
E7663_class.csv   # classification of all contigs (contigs <100bp are not classified)
E7663_class_graph.png   # drawing of the assembly graph with classified contigs



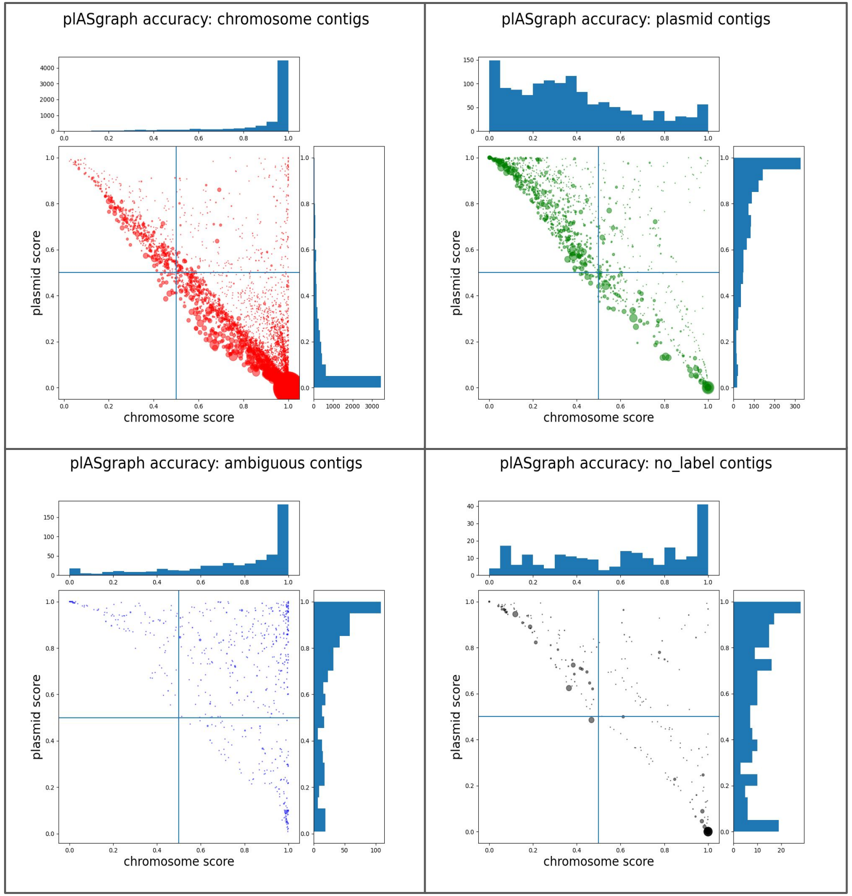
```
contig_name,plasmid_score,chromosome_score,predicted_label
85,0.6056302,0.4807373,Plasmid
139,0.7001808,0.42455834,Plasmid
113,0.6461158,0.8750684,Ambiguous
166,0.9146132,0.71060896,Ambiguous
182,0.89269704,0.25426266,Plasmid
145,0.4363576,0.50318307,Chromosome
1,1.8625138e-05,0.9999825,Chromosome
50,0.03243088,0.9710433,Chromosome
59,0.50970995,0.5564214,Ambiguous
122,0.431104,0.57582986,Chromosome
132,0.53534764,0.5164108,Ambiguous
168,0.09198314,0.95591223,Chromosome
73,0.68665946,0.40148407,Plasmid
```



Results: plASgraph accuracy and vs. PlasForest

Evaluated against a species-specific model, mlplasmid (2018), and a reference-based species-agnostic model, PlasForest (2021), plASgraph outperforms both models (see paper).

Right. Accuracy statistics on a species plASgraph was not trained on, *C. freundii* (96 samples), per category of contig. Each dot is a contig (9118 contigs), with size proportional to the contig size.



plASgraph accuracy: chromosome contigs

plASgraph accuracy: plasmid contigs

plASgraph accuracy: ambiguous contigs

plASgraph accuracy: no_label contigs

Conclusion

Two tools for plasmid contigs classification and binning.
Experimental results suggest a good accuracy.
Developed with no deep knowledge of plasmids biology (e.g. plasmid-specific genes).
Future work: integrate more plasmid knowledge, combine tools.

Acknowledgments

HyAsP: Robert Mueller (Bielefeld University)
PlasBin: Aniket Mane (SFU), Mahsa Faizrahnemoon (SFU)
plASgraph: Janik Sielemann (Bielefeld University), Katharina Sielemann (Bielefeld University), Brona Brejova (Comenius University), Tomas Vinar (Comenius University)
Talk preparation: Aniket Mane (SFU)
Funding: ARETE, NSERC, SFU, PANGAIA (EU).