Reproducing Analysis of MetaGeneMarkS

Karl Gemayel

Sep 23 2020

Contents

1 Introduction

This document serves as a step-by-step instruction manual on how to replicate results from the MetaGeneMarkS paper.

2 Downloading and installing

2.1 Code

Downloading the code is fairly straightforward using `git`. The latest version can be downloaded from `WEBSITE`. To install, simply run

```
source config.sh
cd install; source install.sh; cd ../
```

2.2 Compiling MetaGeneMarkS locally

Running MetaGeneMarkS using automatic genetic code detection is done through the `run_mgm.pl` script found in `$code/hmm_src`. The below compiles the C++ binary and copies all the relevant components to `$bin_external/mgm2_auto`.

```
cd $code/hmm_src;

pf_makefile=Makefile.macos    # NOTE: change based on operating system
make -f $pf_makefile

mkdir -p $bin_external/mgm2_auto
cp gmhmmp2 mgm2_11.mod mgm2_4.mod run_mgm.pl $bin_external/mgm2_auto

mkdir -p $bin_external/mgm2
cp gmhmmp2 mgm2_11.mod mgm2_4.mod $bin_external/mgm2
```

The first command loads all environment variables (including paths to data directories, binaries, etc...), and the second command creates an executable from all python files and stores them in `$bin` for easy access. For non-unix users, you can find the python driver files in `$driverpython`.

2.3 External Tools

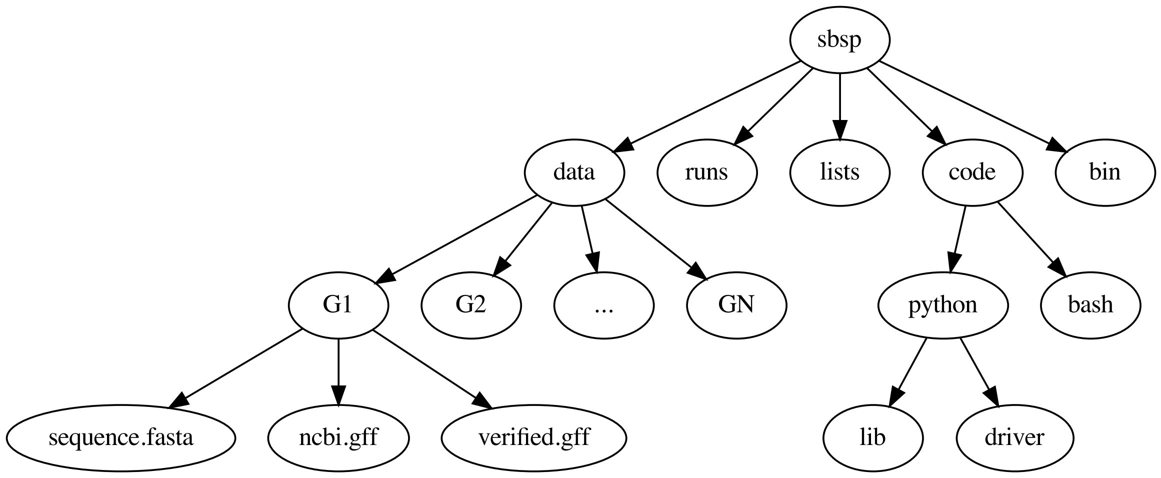
MetaGeneMarkS and MetaGeneMarkS+ and their analysis rely on a handful of external tools. The following need to be installed:

- GeneMarkS-2
  - Used for building MetaGeneMarkS models, and for analysis
  - Link: `http://exon.gatech.edu/GeneMark/license_download.cgi`
- Prodigal:
  - Key competitor for metagenome predictions
  - Link: `https://github.com/hyattpd/Prodigal`
- Other tools:
  - FragGeneScan, MetaGeneAnnotator, MetaGeneMark

2.4 Data

3 Code and data structure

After installing MetaGeneMarkS, you will have the following structure:

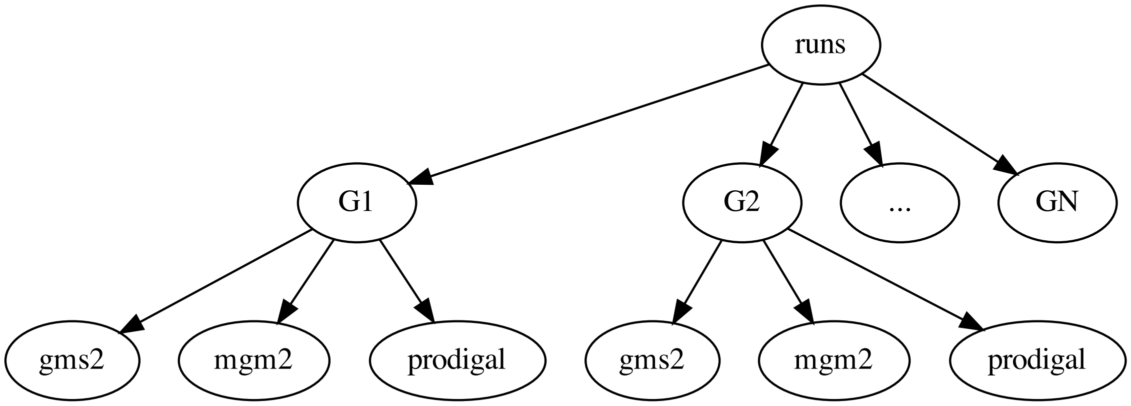


The `bin` directory contains all executables related to MetaGeneMarkS, while the `bin_external` may contain external tools, such as GeneMarkS-2 or Prodigal.

The `data` directory will contain raw genome files (sequence and annotation labels) downloaded from NCBI. In particular, upon initial download of the code, it should contain the genomic sequences for the genomes with experimentally verified gene-starts.

The `list` directory has files that contain different lists of genomes (for example, those with verified genes, those selected as NCBI query genomes, etc. . .)

Finally the `runs` directory will contain runs of different tools, such as MetaGeneMarkS, GeneMarkS-2, or Prodigal (as well as one for NCBI's `PGAP`). These will be placed in a subdirectory per genome, as shown below.



4 Setting up

4.1 Important: Python environment

Scripts to build and analyze results rely on a handful of python packages. The recommended way to install them is to use the `conda` package manager, and simply run

```
conda env create -f install/conda_mgm2.yaml
```

To activate this python environment, run

```
conda activate mg-starts
```

This automatically loads the correct python libraries and executables into `$PATH`.

5 Experiments

5.1 Building MGMS start models

MetaGeneMarkS models are based on thousands of GeneMarkS-2 models derived from a large set of representative genomes. These models have been extracted and ‘pickled‘ together and stored in `$base/raw_models/`. This section provides the script that builds the MetaGeneMarkS models from this set of models.

```
mkdir $tmp/build_models
cd $tmp/build_models

# collect model information
pf_mods_bac=$base/raw_models/bac.pkl
pf_mods_arc=$base/raw_models/arc.pkl

# build models
pf_mgm2_11=mgm2_11.mod
pf_mgm2_4=mgm2_4.mod

pf_mgm_11=$bin_external/gms2/mgm_11.mod
pf_mgm_4=$bin_external/gms2/mgm_4.mod
```

```
# build models for genetic code 11
$bin/build_mgm_models_from_gms2_models_py.sh --pf-bac $pf_mods_bac --pf-arc $pf_mods_arc
    --pf-output $pf_mgm2_11 --pf-mgm $pf_mgm_11 --components "Start Codons" RBS Promoter
    "Start Context" -l INFO --pf-learn-from-options $config/learn_from_6.conf

$bin/build_mgm_models_from_gms2_models_py.sh --pf-bac $pf_mods_bac --pf-arc $pf_mods_arc
    --pf-output $pf_mgm2_4 --pf-mgm $pf_mgm_4 --components "Start Codons" RBS Promoter
    "Start Context" -l INFO --pf-learn-from-options $config/learn_from_6.conf

# These model files need to be placed in the MGM2 binary directory; this download already
    has them in place, so nothing needs to be done.
cd $base
```

5.2 Extract NCBI Protein Homology Predictions

Since part of the analysis uses the protein homology predictions in RefSeq annotation, we separate it out from the annotation files.

```
for pf_gil in $lists/sbsp.list $lists/verified.list; do

  awk -F "," '{if (NR > 1 && NF) print $1}' $pf_gil  | while read -r gcfid; do
    mkdir -p $runs/$gcfid/ncbi_ph;
    mkdir -p $runs/$gcfid/ncbi;

    if [[ -f $data/$gcfid/verified.gff ]]; then
      mkdir -p $runs/$gcfid/verified;
      cp $data/$gcfid/verified.gff $runs/$gcfid/verified/prediction.gff;
    fi

    awk -F "\t" '{if ($2 == "Protein Homology") print }' $data/$gcfid/ncbi.gff >
        $runs/$gcfid/ncbi_ph/prediction.gff;
    cp $data/$gcfid/ncbi.gff $runs/$gcfid/ncbi/prediction.gff
  done
done
```

5.3 Complete Genomes

This section starts with “setup” code to link to directories, parallelization options, and the tools for experimenting. Please read through to identify any parameters that you would like to change (nothing needs to be changed to reproduce the results shown in the paper).

```
function init_experiment() {
  local dn_experiment="$1"
  cd $tmp
  mkdir -p $dn_experiment
  cd $dn_experiment
}

pf_mgm_mod=$bin_external/mgm/MetaGeneMark_v1.mod
pf_mgm2_mod=$bin_external/mgm2_auto/mgm2_11.mod
pf_prl_options=$config/parallelization_pbs_2.conf

declare -a tools=(mgm fgs mga mprodigal mgm2_auto prodigal gms2);
declare -a mgtools=(mgm fgs mga mprodigal mgm2_auto);

# Setting this to empty removes PBS parallelization.
toggle_pbs=""
# toggle_pbs="--pf-parallelization-options $pf_prl_options"

function run_on_complete() {
  local pf_gil="$1"
  for t in "${tools[@]}"; do
    $bin/run_tool_on_genome_list_py.sh --pf-gil $pf_gil --type auto --tool $t --pf-mgm-mod
        $pf_mgm_mod --pf-mgm2-mod $pf_mgm2_mod ${toggle_pbs}
  done
}

function collect_stats_complete() {
  local pf_gil="$1"
  local pf_stats="$2"
  local refs="$3"
  $bin/stats_per_gene_py.sh --pf-gil $pf_gil --tools $refs "${tools[@]}"  --pf-output
      $pf_stats ${toggle_pbs}
}
```

5.3.1 Verified Starts

```
# run tools on verified
dn_experiment=complete_verified
init_experiment $dn_experiment

pf_gil=$lists/verified.list
run_on_complete "$pf_gil"

# collect statistics
pf_stats=$(pwd)/summary_complete_verified.csv
collect_stats_complete $pf_gil $pf_stats "verified ncbi ncbi_ph"

# visualize statistics
mkdir -p figures
cd figures
$bin/viz_stats_small_py.sh --pf-data $pf_stats --ref-3p ncbi --ref-5p verified --tools
    "${tools[@]}"

#back to start
cd $base
```

5.3.2 StartLink+ Starts

```
# run tools on startlink
dn_experiment=complete_startlink
init_experiment $dn_experiment

pf_gil=$lists/sbsp.list
run_on_complete "$pf_gil"
```

```
# collect statistics
pf_stats=$(pwd)/summary_complete_startlink.csv
collect_stats_complete $pf_gil $pf_stats "sbsp ncbi ncbi_ph"

# visualize statistics
mkdir -p figures
cd figures
$bin/viz_stats_large_py.sh --pf-data $pf_stats --ref-3p ncbi --ref-5p sbsp ncbi_ph --tools
    mgm fgs mga mprodigal mgm2 --pf-checkpoint-3p checkpoint_3p.pkl --pf-checkpoint-5p
    checkpoint_5p.pkl

# back to start
cd $base
```

5.4 Genome Fragments

```
function run_on_chunks() {
  local pf_gil="$1"
  local pf_runs_summary="$2"

  $bin/run_tools_on_chunks_py.sh --pf-gil $pf_gil --tools "${mgtools[@]}" --pf-mgm2-mod
      $pf_mgm2_mod --pf-mgm-mod $pf_mgm_mod --pd-work $runs --pf-summary $pf_runs_summary
      ${toggle_pbs}

}
```

5.4.1 Verified Starts

```
dn_experiment=chunks_verified
init_experiment $dn_experiment

pf_gil=$lists/verified.list

# run on chunks
chunk_sizes="250 500 750 1000 1250 1500 1750 2000 2250 2500 2750 3000 5000"
pf_runs_summary=$(pwd)/runs_summary_chunks_verified.csv
$bin/run_tools_on_chunks_py.sh --pf-gil $pf_gil --tools "${tools[@]}" --pf-mgm2-mod
    $pf_mgm2_mod --pf-mgm-mod $pf_mgm_mod --pd-work $runs --pf-summary $pf_runs_summary
    --chunk-sizes-nt $chunk_sizes ${toggle_pbs}



# collect statistics
pf_stats=$(pwd)/summary_chunks_verified.csv
$bin/stats_per_gene_on_chunk_py.sh --pf-summary $pf_runs_summary  --reference-tools
    verified ncbi  --pf-output $pf_stats ${toggle_pbs}

# visualize statistics
mkdir -p figures
cd figures
```

```
$bin/viz_stats_per_gene_on_chunks_py.sh --pf-data $pf_stats --ref-3p ncbi --ref-5p verified
    --tools "${tools[@]}"

cd $base
```

5.4.2 StartLink Starts

```
dn_experiment=chunks_startlink
init_experiment $dn_experiment

pf_gil=$lists/sbsp.list

pf_runs_summary=$(pwd)/runs_summary_chunks_startlink.csv
run_on_chunks $pf_gil $pf_runs_summary


# collect statistics
pf_stats=$(pwd)/summary_chunks_startlink.csv
#$bin/stats_per_gene_on_chunk_py.sh --pf-summary $pf_runs_summary  --reference-tools sbsp
    ncbi_ph ncbi  --pf-parallelization-options $pf_prl_options --pf-output $pf_stats
$bin/stats_per_gene_on_chunk_py.sh --pf-summary $pf_runs_summary  --reference-tools ncbi
    sbsp  ncbi_ph  --pf-output $pf_stats --batch-size 40 ${toggle_pbs}

# visualize statistics
mkdir -p figures
cd figures

 #      $bin/viz_stats_per_gene_on_chunks_py.sh --pf-data $pf_stats --ref-3p ncbi --ref-5p
     sbsp ncbi_ph --tools "${mgtools[@]}"
$bin/viz_stats_per_gene_on_chunks_large_py.sh --pf-data $pf_stats --ref-5p sbsp ncbi_ph
    --ref-3p ncbi --tools "${mgtools[@]}"   --pf-checkpoint-5p checkpoint_5p.pkl
    --pf-checkpoint-3p checkpoint_3p.pkl ${toggle_pbs}

cd $base
```