PGAP and RAPT hands-on activities

PGAP

In this section you will:

1. set up a virtual machine
2. download the PGAP Docker image
3. annotate a genome
4. interpret PGAP outputs

References

The PGAP wiki: https://github.com/ncbi/pgap/wiki

- Provides the PGAP command line interface
- Describes the PGAP inputs and outputs

PGAP prep, from getting a machine to getting the software

Create a virtual machine (VM)

Log in to an authorized account and go to https://console.cloud.google.com/home/dashboard?project=ncgas-cloud-workshop

Click "Compute Engine" or navigate to the "Compute Engine" section by clicking on the navigation menu  on the top left corner.

Click on the blue “Create instance” button  on the top bar.

Create an image with the following parameters: (if parameter is not list below, keep the default setting)

- Name: use your name in the instance, something like: <lastname>-pgap
- Region: us-central1 (Iowa)
- Machine family: General Purpose
- Series: N1
- Machine type: n1-highmem-16
- Boot Disk:
    - Click "Change,"
    - Go to the Custom Image tab
    - In the "Show images from" drop-down select "ncgas-cloud-workshop"
    - In the "Images" drop-down, select "ncgas-custom-pgap-bootdisk"
    - Set the boot disc size to 250 GB
    - Click "Select"
- Click the blue “Create” button. This will create and start the VM.

Once you have your VM created (should take a couple of minutes), you must SSH into it. There are many methods to access your VM, depending on the ways in which you would like to use it. On the GCP Console, the most straightforward way is to SSH from the browser. To do this, click your instance to open its view and then click the SSH button to open a popup terminal.

Download the PGAP command line interface, and Docker image

- Download the PGAP command line interface

```
curl -OL https://github.com/ncbi/pgap/raw/prod/scripts/pgap.py
chmod +x pgap.py
```

- Download the Docker image

```
./pgap.py --update --taxcheck
```

Long! This can take up to 30 minutes. While the download and extraction is happening, skip to the RAPT section below

Run PGAP on a full Klebsiella pneumonia genome

Once the PGAP image and reference data is downloaded, you are ready to run PGAP. We have pre-loaded the VM with some of the input files needed to run the exercises below, but not all.

The fasta sequences for the Klebsiella pneumonia genome you are going to annotate are provided in /pgap_data/K_pneu2.fasta, but the YAML files are not.

Prepare the YAML files

Important note about YAML: use spaces to indent (NOT tabs)

- Generic YAML file:

This file provides the location of the fasta and of the metadata to the program. You can create it this way from the command line:

```
cat << EOF_input > pgap_data/K_pneu2.yaml
fasta:
    class: File
    location: K_pneu2.fasta
submol:
    class: File
    location: K_pneu_meta2.yaml
EOF_input
```

- Metadata YAML file

This file provides the metadata associated with the fasta file. At a minimum, it should contain the genus. More information should be provided if you intend to submit the resulting annotation to GenBank. See a full description of the inputs consumed by PGAP and whether they are required for submission to GenBank at https://github.com/ncbi/pgap/wiki/Input-Files. See also pgap_data/K_pneu_meta.yaml for an example.

In this exercise, for simplicity, just provide the genus species in the metadata file.

```
cat << EOF_input > pgap_data/K_pneu_meta2.yaml
organism:
    genus_species: 'Klebsiella pneumoniae'
EOF_input
```

Run PGAP

Now you are ready to run PGAP on this Klebsiella genome.

Use `nohup` to protect yourself from loss of the SSH connection:

```
nohup ./pgap.py -n -o K_pneu2_annot pgap_data/K_pneu2.yaml &
disown %1
```

This execution will take several hours. Check the results on Day 2!! You can follow the progress by consulting the cwltool.log in the output directory. In the meantime, for a sneak preview of the results, see just below.

Review the results

We have pre-computed results for the genome you just started annotating, as well as another K. pneumoniae genome. You can find them in pgap_data as compressed directories. Decompress them with:

```
cd pgap_data
tar -xzvf K_pneu_annot_pre.tar.gz
tar -xzvf K_pneu2_annot_pre.tar.gz
```

The description of the outputs is in https://github.com/ncbi/pgap/wiki/Output-Files

Compare the annotation results for these two genomes. Any noticeable differences?

Taxonomy verification

You may not be certain of the taxonomy of the genome you have sequenced and assembled.

This is not an issue! You can use `--taxcheck` or `--taxcheck-only` to verify your sample prior to running PGAP.

Let's try an example.

- Prepare the input using your best guess for the genus or genus species of the assembled genome

In this case, you happen to think that the assembled genome you wish to annotate (K_pneu2.fasta) is from a Salmonella enterica strain, so you'd make a metadata YAML like:

```
cat << EOF_input > pgap_data/Sent_meta.yaml
organism:
    genus_species: 'Salmonella enterica'
EOF_input
```

And the generic YAML:

```
cat << EOF_input > pgap_data/Sent.yaml
fasta:
    class: File
    location: K_pneu2.fasta
submol:
    class: File
    location: Sent_meta.yaml
EOF_input
```

- Run the taxonomy assignment verification:

To verify only, without attempting PGAP afterward, use `--taxcheck-only`

```
./pgap.py -n --taxcheck-only -o Sent_out_TConly pgap_data/Sent.yaml
```

To verify and run PGAP afterward **if** the genus is confirmed, use `--taxcheck`

```
./pgap.py -n --taxcheck -o Sent_out pgap_data/Sent.yaml
```

- Check the results (note: documentation for the outputs of the taxonomy check is in: https://github.com/ncbi/pgap/wiki/Taxonomy-Check)

Guess the issue #1

Let's try running PGAP on case guess1, using the guess1 inputs already available in pgap_data:

```
./pgap.py -n --taxcheck -o guess1_out pgap_data/guess1.yaml
```

Review the results in the output directory. Can you find out why the process stopped?

Guess the issue #2

Run PGAP using the guess2 inputs already available in pgap_data:

```
./pgap.py --taxcheck -n -o guess2_out pgap_data/guess2.yaml
```

Review the results in the output directory. Review the results in the output directory. Why did the process stop?

How would you annotate a plasmid?

Given the plasmid sequence plasmid.fasta, in pgap_data, isolated from a Salmonella enterica strain, how would you annotate it?

Create the YAML files and determine what options will allow the annotation to finish successfully.

Hint: mentioned at the end of the intro presentation on Day 1...

Take-aways on PGAP options

`--taxcheck-only`: use to verify that the genus you assigned to the sequenced genome.. Does not run PGAP!

`--taxcheck:` use to verify the genus and run PGAP afterward, only if the genus is confirmed

`--ignore-all-errors`: use if you do not wish PGAP to stop because of vector or adapter contamination, or because the genome is smaller or larger than expected for this species, or because there are sequences with training Ns, or sequences that are shorter than 200 bases in the input

`--taxcheck --ignore-all-errors`: use to use to verify the genus and run PGAP afterward, whether or not the genus is confirm and whether or not there are other issues in the input sequences

RAPT

In this section you will

1. run RAPT in GCP using fasta or a run in SRA as input
2. interpret the RAPT outputs
3. learn about stand-alone RAPT, for non GCP users

References

RAPT github: https://github.com/ncbi/rapt

- Provides the RAPT command line interface for GCP RAPT and stand-alone RAPT
- Also has links to PGAP and SKESA publications and github sites

Download the RAPT command line interface

Go to https://console.cloud.google.com/home/dashboard?project=ncgas-cloud-workshop

Open a Cloud Shell (NO NEED for a VM) by clicking the Activate Cloud Shell button  .

Download the RAPT command line interface:

```
curl -sSLo rapt.tar.gz https://github.com/ncbi/rapt/releases/download/v0.2.0/rapt-v0.2.0.tar.gz
tar -xzf rapt.tar.gz && rm -f rapt.tar.gz
```

Run RAPT

Starting from a SRR run

Using SRR8602029, https://www.ncbi.nlm.nih.gov/sra/?term=SRR8602029. This a Staphylococcus aureus paired end Illumina run.

Run the command, after substituting your bucket name (gs://...) in the command:

```
./run_rapt_gcp.sh submitacc SRR8602029 -b <yourbucket> --label SRR8602029_sra --regions us-central1
```

(You may get a window pop up with the text: "Authorize Cloud Shell - gcloud is requesting your credentials to make a GCP API call. Click to authorize this and future calls that require your credentials." Click Authorize.)

By specifying `--regions us-central1`, you ensure that the VM that will be created by the command is in region `us-central1` (if not specified, the VM will be created in `us-east4`)

Starting from a fasta file of Illumina reads in a bucket

Check the fasta file

```
gsutil cat gs://pgap-rapt-material/rapt_data/SRR8602029.fastq | grep @ | head
```

You will noticed that the two reads for each spots are adjacent in the file. This is a requirement for correct handling of paired ends by SKESA.

Run the command, after substituting your bucket name (gs://...) in the command:

```
./run_rapt_gcp.sh submitfastq gs://pgap-rapt-material/rapt_data/SRR8602029.fastq --organism "Staphylococcus
aureus" -b <yourbucket> --label SRR8602029_fq --regions us-central1
```

Track jobs

These jobs will take about two hours.

- You can check that the status of jobs you have launched with the `joblist` command (pipe to grep <yourname> to see only yours)

```
./run_rapt_gcp.sh joblist
```

- You can also follow the progress of a running job in the Log Viewer

Replace <jobid> with the jobid for the execution you wish to track, and copy in your browser:

https://console.cloud.google.com/logs/viewer?project=ncgas-cloud-workshop&filters=text:<jobid>

Review the results

Once done, copy the output to the Google terminal for both jobs, and untar:

```
gsutil cp -r <yourbucket>/<jobid> .
cd <jobid>
tar -xzvf output.tar.gz
```

Compare the results from both RAPT executions.

Are they the same?

Extras

Would like to run RAPT outside of this workshop but don't have a GCP account?

See https://github.com/ncbi/rapt/blob/master/Standalone%20RAPT.md