

# NCBI tools for microbial genomics

Françoise Thibaud-Nissen, PhD

November 16, 2020



U.S. National Library of Medicine  
*National Center for Biotechnology Information*

# Instructor information

- Françoise Thibaud-Nissen, PhD
- Plant biologist by training
- Team Lead for RefSeq Prokaryotes at the NCBI/NLM/NIH



# Outline

- Annotation in the biological analysis workflow
- What is the stand-alone Prokaryotic Genome Annotation Pipeline (PGAP)?
- Start-to-finish annotation with PGAP
- The Read Assembly and Annotation Pipeline Tool (RAPT) = assembly + annotation
- Start-to-finish assembly and annotation with RAPT

<https://github.com/ncbi/pgap>

<https://github.com/ncbi/rapt>

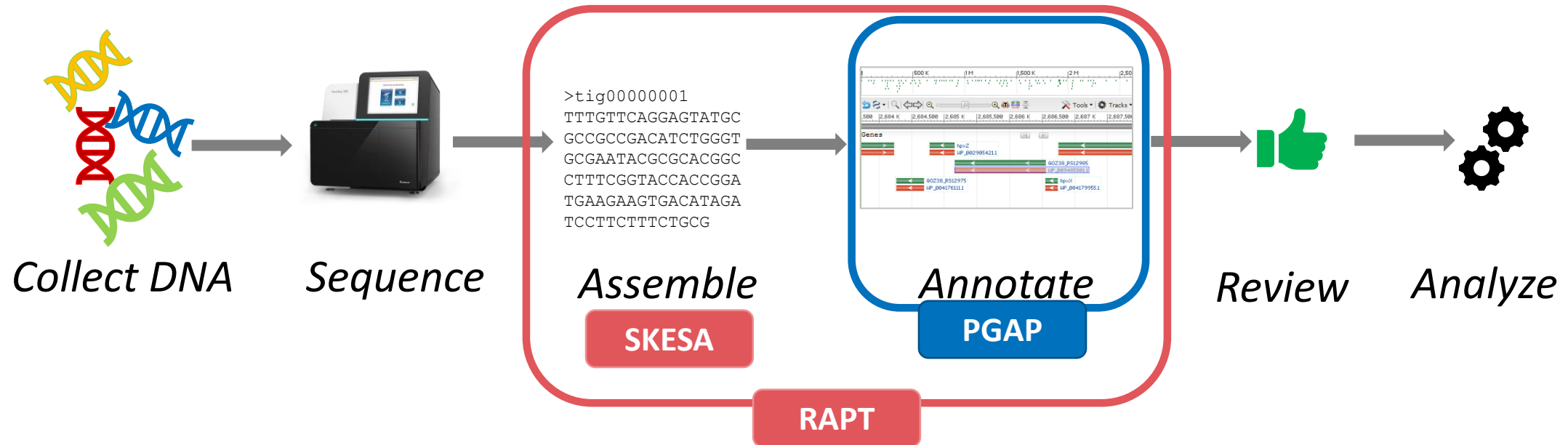


# Importance of gene annotation in whole-genome analyses

- Determination of gene location and function
- Precursor of downstream analyses, including
  - Comparative genomics
  - Phylogenetic trees
  - Exploration of biochemical pathways
  - Virulence factor discovery
  - ...
- Assessment of sample or assembly quality



# PGAP and RAPT in a typical workflow

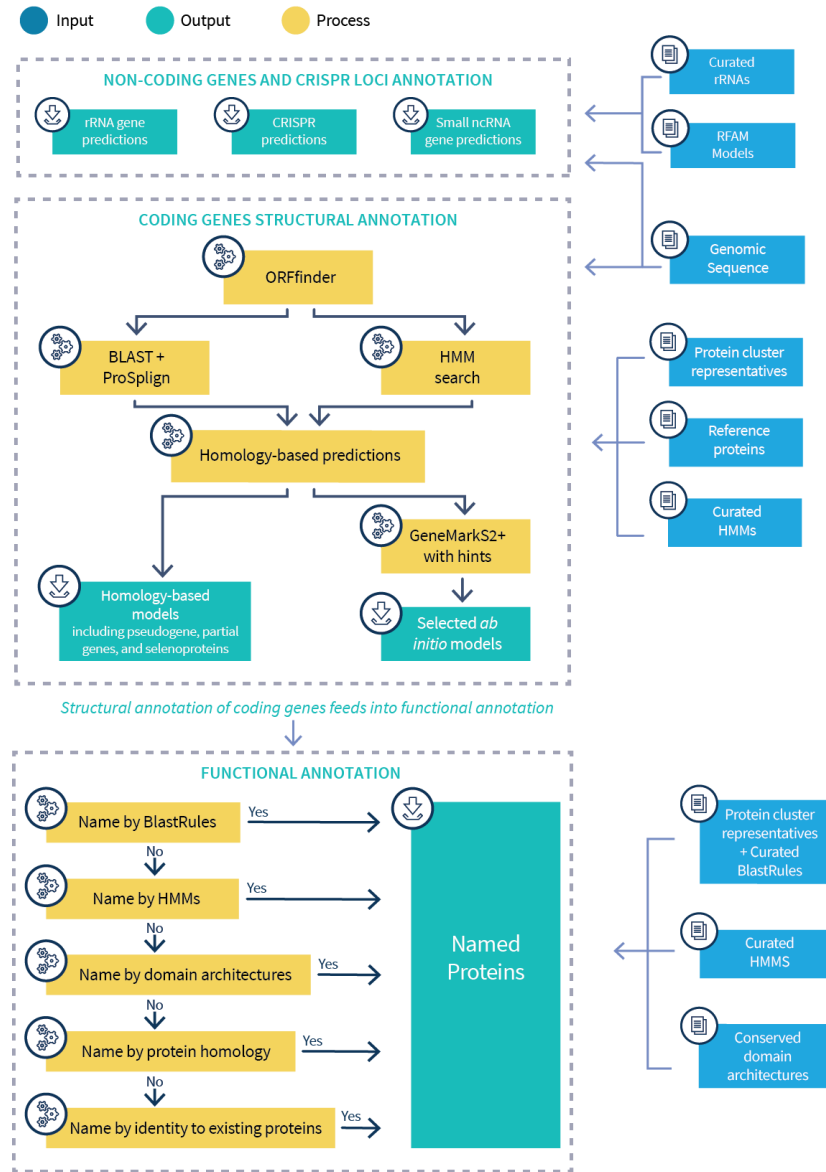


## What does PGAP do?

- Annotates the genome with PGAP, the pipeline used to annotate RefSeq genomes

## What does RAPT do?

- Assembles a genome from **Illumina** short reads sequenced from a bacterial or archaeal isolate, using SKESA
- Annotates the genome with PGAP, the pipeline used to annotate RefSeq genomes



# PGAP in brief

## RefSeq annotation pipeline

- Automated pipeline
- Protein-coding gene prediction with:
  - Protein homology
  - Ab initio calls (GeneMarkS2+)
  - Protein profile hidden Markov models
- Non-coding gene prediction with:
  - tRNAscan-SE
  - RFAM
- Functional annotation with:
  - hidden Markov models
  - BlastRules
  - CDD architectures
  - Protein homology

# Why run PGAP yourself?

- Proprietary genomes
- Quick characterization of starting material
- Assembly QC
- Large volume
- Comparison to PGAP-annotated RefSeq assemblies

# What is stand-alone PGAP?

- Our goal
- PGAP releases
- Systems requirements

## How is stand-alone PGAP packaged?

A PGAP release is:

- A Docker image in dockerhub containing:
  - The pipeline in CWL (= the “glue” between the applications)
  - The binaries (pgap-utils)
  - cwltool, the reference runner for CWL
- The reference data on AWS
  - Custom Blast databases
  - HMM models
  - Taxonomy database
  - etc...



# What is stand-alone PGAP?

- How does it work?
- PGAP releases
- Systems requirements

## Hardware and software requirements

- Linux, Windows, Mac OSX
- Python (version 3.6 or higher)
- Docker, Singularity or podman
- 100 GB disk free space
- 8-core CPU, 32 GB memory recommended

# Start-to-finish PGAP annotation

- Download pgap.py
- Download the Docker image and reference data
- Prepare the inputs
  - Input file format
  - FASTA
  - Metadata
- Run the annotation in the container
- Review the results

## pgap.py, the PGAP command line interface

- Choose where to run PGAP
  - Locally
  - On a remote machine
  - In the cloud
- Download the convenience script, pgap.py

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

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current	
			Dload	Upload	Total	Spent	Left	Speed
100	130	100	130	0	0	1203	0	--:--:-- --:--:-- --:--:-- 1203
100	24389	100	24389	0	0	187k	0	--:--:-- --:--:-- --:--:-- 187k

- Make pgap.py executable

```
chmod +x pgap.py
```

# Start-to-finish PGAP annotation

- Download the pgap.py
- Download the Docker image and reference data
- Prepare the inputs
  - Input file format
  - FASTA
  - Metadata
- Run the annotation in the container
- Review the results

## Download the Docker image and the reference data

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

```
The latest version of PGAP is 2020-09-24.build4894, you have nothing  
installed locally.  
Downloading (as needed) Docker image ncbi/pgap:2020-09-24.build4894  
2020-09-24.build4894: Pulling from ncbi/pgap  
Digest:  
sha256:d2d57c18a3cbcf51179b17c34f349752737edbd21edca9a9a078e5d181c42008  
Status: Image is up to date for ncbi/pgap: 2020-09-24.build4894  
Installing PGAP reference data version 2020-09-24.build4894  
Downloading and extracting tarball:  
https://s3.amazonaws.com/pgap/input-2020-09-24.build4894.tgz  
Downloaded 16485112395 of 16485112395 bytes (100.00%)
```

### Reference data:

- input-2020-09-24.build4894.tgz

### Example genomes (including *Mycoplasma genitalium* G37)

- test\_genomes-2020-09-24.build4894.tgz

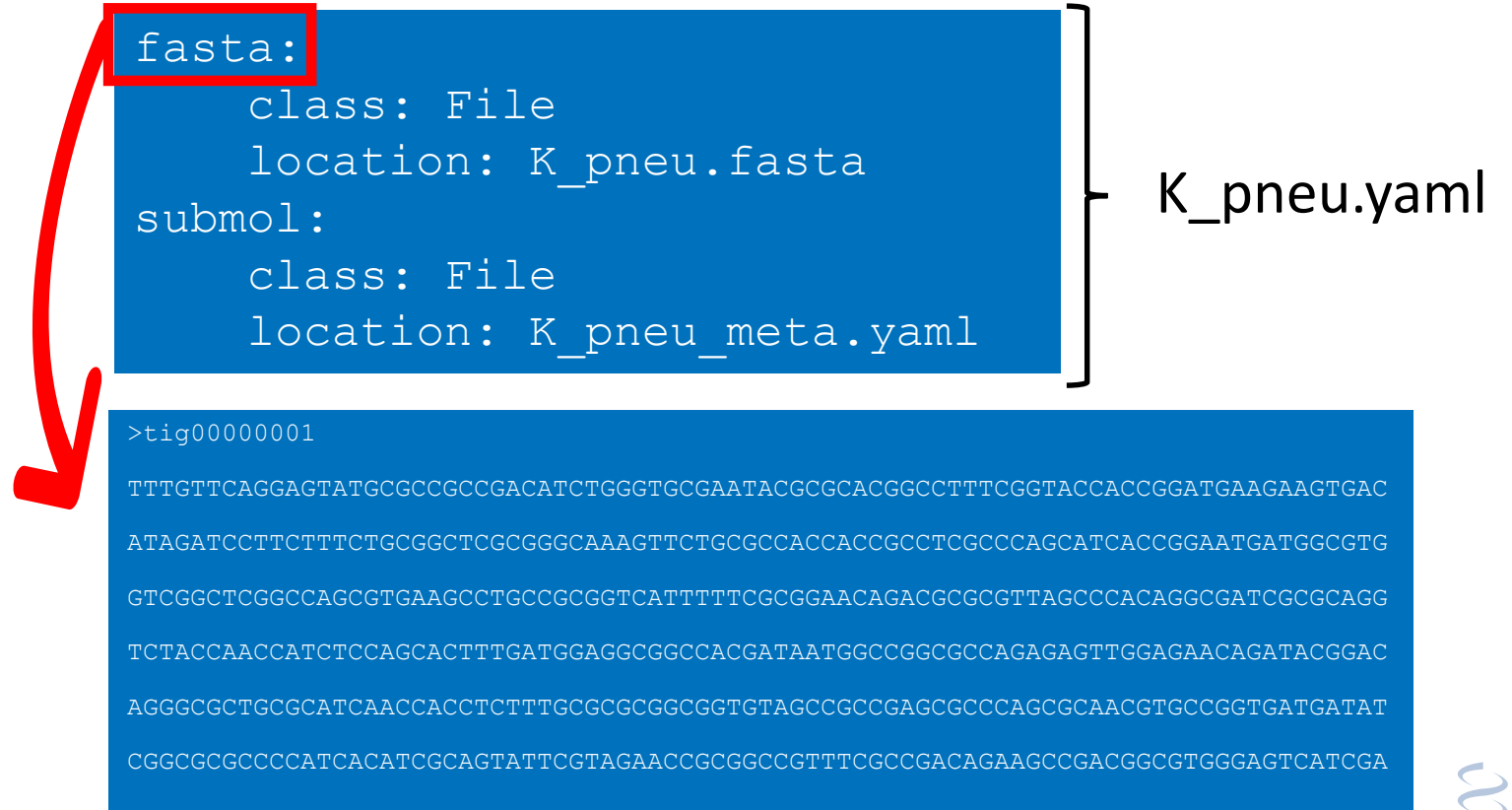
# Start-to-finish PGAP annotation

- Download pgap.py
- Download the Docker image and reference data
- Prepare the inputs
  - Generic YAML
  - FASTA
  - Metadata
- Run the annotation in the container
- Review the results

## Let's annotate a *Klebsiella pneumoniae* genome

The generic YAML file: a pointer to two other files:

- Multifasta of the assembly
- Metadata (organism, contact info, author(s))



# Start-to-finish PGAP annotation

- Download pgap.py
- Download the Docker image and reference data
- Prepare the inputs
  - Input file format
  - FASTA
  - Metadata
- Run the annotation in the container
- Review the results

## Generic YAML file: a pointer to two other files

```
fasta:  
  class: File  
  location: K_pneu.fasta  
submol:  
  class: File  
  location: K_pneu_meta.yaml
```

K\_pneu.yaml

```
topology: 'linear'  
organism:  
  genus_species: 'Klebsiella pneumoniae'  
contact_info:  
  last_name: 'Thibaud-Nissen'  
  first_name: 'Francoise'  
  email: 'thibaudf@ncbi.nlm.nih.gov'  
  organization: 'NCBI prok group'  
  department: 'Department of Microbiology'  
  street: '9000 Rockville Pike'  
  city: 'Bethesda'  
  postal_code: '20845'  
  state: 'Maryland'  
  country: 'USA'  
authors:  
- author:  
  first_name: 'Peter'  
  last_name: 'Cooper'
```

Required only if  
submitting to  
GenBank

# Start-to-finish PGAP annotation

- Download the pgap.py
- Download the Docker image and reference data
- Prepare the inputs
  - Input file format
  - FASTA
  - Metadata
- Run the annotation in the container
- Review the results

## Annotate your genome

```
$ ./pgap.py -r -o K_pneu_results K_pneu.yaml
PGAP version 2020-09-24.build4894 is up to date.
Output will be placed in: /home/thibaudf/K_pneu_results
PGAP completed successfully.
```

- r : allows transmitting to NCBI that the execution has started and stopped
- o : path to the output directory



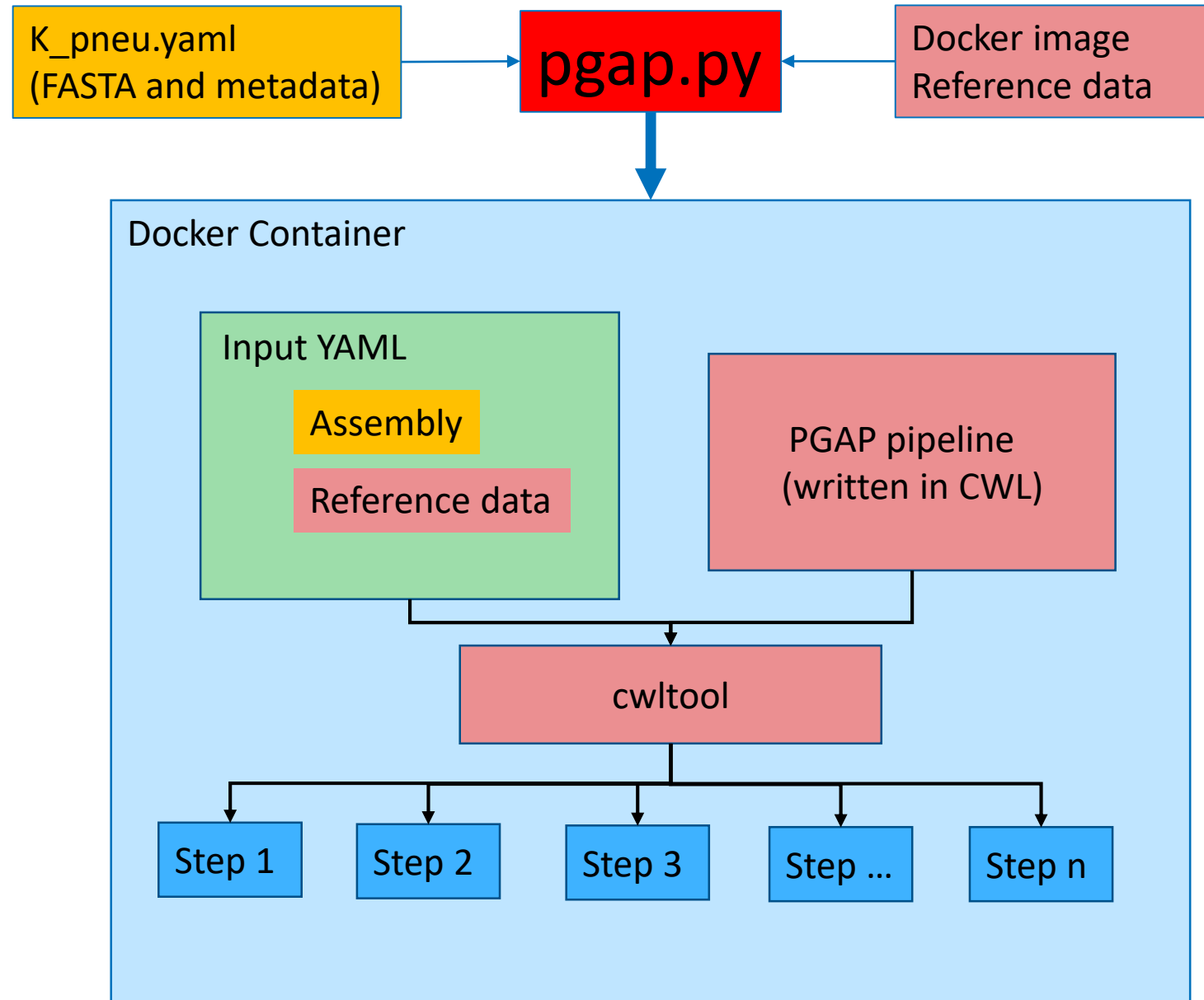
Tip: use screen, tmux or nohup if running remotely

```
$ nohup ./pgap.py -r -o K_pneu_results K_pneu.yaml &
$ disown <pid>
```

# Start-to-finish PGAP annotation

- Download the `pgap.py`
- Download the Docker image and reference data
- Prepare the inputs
  - Input file format
  - FASTA
  - Metadata
- Run the annotation in the container
- Review the results

## Execution of PGAP in a container



# Review the results

- Output formats
  - FASTA
  - GFF
  - ASN.1
  - GenBank
- Metadata
  - Topology
  - Species
  - Submitter
  - Submitter info
- Annotation summary
  - Run information
  - Many frameshifts!

## Output

- The annotation in a variety of formats
  - annot.faa – annotated proteins (fasta format)
  - annot.sqn – annotated genomic sequence in ASN format (use for submission)
  - annot.gff – annotated genomic sequence in GFF3 format
  - annot.gbk – annotated genomic sequence in GenBank flat file format
- Vector or adaptor sequences in tab-delimited format
  - calls.tab: Coordinates of vector or adaptor sequences in tab-delimited format.
- (Taxonomic assignment verification files. Only produced if using the flag --taxcheck or --taxcheck-only)
  - ani-tax-report.txt: Results of the taxonomy check in text format.
  - ani-tax-report.xml: Results of the taxonomy check in xml format.

More about input QA in the hands-on activities!





# Run PGAP on draft assemblies

## “Pre-annotation”

Get a sense of the assembly quality early in your workflow

With the flag `--ignore-all-errors`

```
./pgap.py -r \  
--ignore-all-errors \  
-o K_pneu_results \  
K_pneu.yaml
```

- Ignores the genome size check
- Ignores invalid sequences due to
  - Vector and adaptor sequences
  - Stretches of Ns at the end of sequences
- Allows undefined species (i.e. *Rickettsia* sp.)



Results may not comply with GenBank quality criteria



# Review the results

- Output formats
  - FASTA
  - GFF
  - ASN.1
  - GenBank
- Metadata
  - Topology
  - Species
  - Submitter
  - Submitter info
- Annotation summary
  - Run information
  - Many frameshifts!

## Info provided on input in the output

```
LOCUS      tig00000001      5364382 bp      DNA      linear      BCT 26-OCT-2020
DEFINITION Klebsiella pneumoniae, whole genome shotgun sequence.
ACCESSION
VERSION
KEYWORDS   WGS.
SOURCE     Klebsiella pneumoniae
           ORGANISM Klebsiella pneumoniae
           Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacterales;
           Enterobacteriaceae; Klebsiella.
REFERENCE  1  (bases 1 to 5364382)
           AUTHORS  Cooper, P.
           TITLE    Direct Submission
           JOURNAL  Submitted (08-DEC-2019) Department for Microbiology, National
           Center for Biotechnology Information, 9000 Rockville Pike,
           Bethesda, Maryland 20845, USA
COMMENT    The annotation was added by the assembly submitters using the NCBI
           Prokaryotic Genome Annotation Pipeline (PGAP). Information about
           stand-alone PGAP can be found here: https://github.com/ncbi/pgap/
```

# Review the results


- Output formats
  - FASTA
  - GFF
  - ASN.1
  - GenBank
- Metadata
  - Topology
  - Species
  - Submitter
  - Submitter info
- Annotation summary
  - Run information
  - Many frameshifts!

## Annotation summary

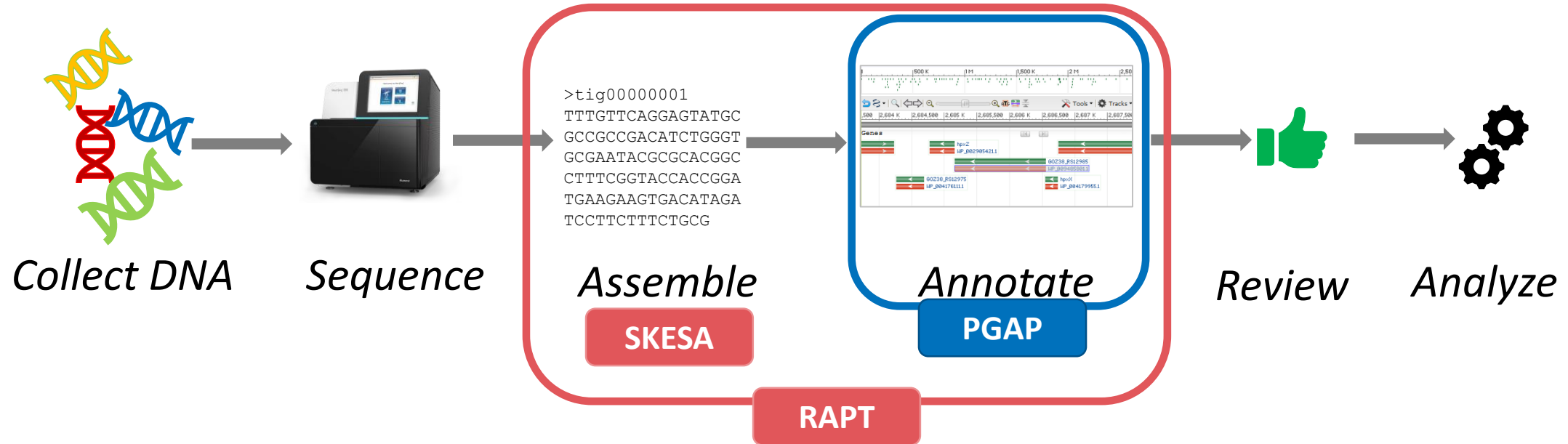
```
##Genome-Annotation-Data-START##
Annotation Provider      :: Organization
Annotation Date         :: 10/26/2020 13:35:31
Annotation Pipeline     :: NCBI Prokaryotic Genome Annotation
                        Pipeline (PGAP)
Annotation Method       :: Best-placed reference protein set;
                        GeneMarkS-2+
Annotation Software revision :: 2020-09-24.build4894
Features Annotated      :: Gene;CDS;rRNA;tRNA;ncRNA;repeat_region
Genes (total)           :: 6,316
CDSs (total)            :: 6,194
Genes (coding)          :: 2,172
CDSs (with protein)     :: 2,172
Genes (RNA)             :: 122
rRNAs                   :: 9, 8, 8 (5S, 16S, 23S)
complete rRNAs          :: 9, 8, 8 (5S, 16S, 23S)
tRNAs                   :: 87
ncRNAs                  :: 10
Pseudo Genes (total)    :: 4,022
CDSs (without protein)  :: 4,022
Pseudo Genes (ambiguous residues) :: 0 of 4,022
Pseudo Genes (frameshifted) :: 3,875 of 4,022
Pseudo Genes (incomplete) :: 252 of 4,022
Pseudo Genes (internal stop) :: 104 of 4,022
Pseudo Genes (multiple problems) :: 205 of 4,022
##Genome-Annotation-Data-END##
```

Run information

Annotation results



# The Read Assembly and Annotation Pipeline Tool (RAPT): An extension of PGAP



## What does PGAP do?

- Annotates the genome with PGAP, the pipeline used to annotate RefSeq genomes

## What does RAPT do?

- Assembles a genome from **Illumina** short reads sequenced from a bacterial or archaeal isolate, using SKESA
- Annotates the genome with PGAP, the pipeline used to annotate RefSeq genomes

# GCP RAPT Workflow Overview

From the Google Console Shell



## Requirements

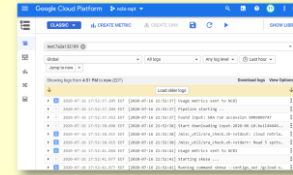
- GCP account and project with a quota of at least 16 CPU in a given region
- Google Life Sciences API enabled
- A Google storage bucket



Starts a virtual machine



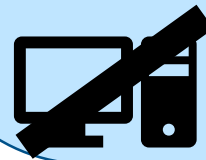
Downloads and instantiates the RAPT Docker container



Sends execution logs to Google Stackdriver



Copies results to Google storage bucket



Shuts down the virtual machine



U.S. National Library of Medicine  
National Center for Biotechnology Information



# GCP RAPT

## Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

## run\_rapt\_gcp.sh, the command line interface for RAPT

Simply copy and paste these two commands at your Cloud Shell prompt, as specified in <https://github.com/ncbi/rapt/>

```
$ 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
```

```
$ ls  
CHANGELOG.md README.txt release-notes.txt  
run_rapt_gcp.sh run_rapt.py
```

# GCP RAPT Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

## Explore GCP RAPT

This command will provide instructions and options for running RAPT:

```
$. /run_rapt_gcp.sh help
./run_rapt_gcp.sh help
Usage: run_rapt_gcp.sh <command> [options]
Job creation commands:
    submitacc <sra_acxn> <-b|--bucket URL> [--label LABEL]
        [--skesa-only] [--no-usage-reporting] [--machine-type TYPE]
        [--boot-disk-size NUM] [--timeout SECONDS]
    Submit a job to run RAPT on an SRA run accession (sra_acxn).
    submitfastq <fastq_uri> <--organism "Genus species"> [--strain "ATCC xxxx"]
        <-b|--bucket URL> [--label LABEL] [--skesa-only]
        [--no-usage-reporting] [--machine-type TYPE] [--boot-disk-size NUM]
        [--timeout SECONDS]
    Submit a job to run RAPT on Illumina reads in FASTQ or FASTA format.
    fastq_uri is expected to point to a google cloud storage (bucket).

    The --organism argument is mandatory. It is the binomial name or, if the
    species is unknown, the genus for the sequenced organism. This identifier
    must be valid in NCBI Taxonomy. The --strain argument is optional.

[...]
```

Common options:

```
=====
-b|--bucket URL
    Mandatory. Specify the destination storage location to store results
    and job logs.
--label LABEL
    Optional. Tag the job with a custom label, which can be used to filter jobs
```

# GCP RAPT

## Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

## Using an SRA run as input to RAPT

```
$ ./run_rapt_gcp.sh submitacc SRR11675939  
-b gs://mybucket --label Mbovis_NADC67
```

**submitacc SRR11675939** - input SRA run  
**-b gs://mybucket** – bucket where output will be copied  
**--label Mbovis\_NADC67** – optional tag



# GCP RAPT

## Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

## Using fastq files as input to RAPT

To run RAPT with a fastq or fasta file, use the `submitfastq` command, point to the location of the fastq file, and use the `-b` option to point to your Google bucket for the output location.

Here is an example of using an input file stored locally:

```
$ ./run_rapt_gcp.sh submitfastq  
./Mp_ATCC_25960_local.fastq --organism "Mycoplasma  
pirum" --strain "ATCC 25960" -b gs://mybucket --  
label Mp_25960_l
```

(Please note that the quotes are required for the organism and strain options.)

Here is an example of using an input file in a Google bucket:

```
$ ./run_rapt_gcp.sh submitfastq  
gs://friendbucket/Mp_ATCC_25960.fastq --organism  
"Mycoplasma pirum" -b gs://mybucket --label  
Mp_25960_b
```

# GCP RAPT

## Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

## Launch the job...

```
$/run_rapt_gcp.sh submitfastq gs://friendbucket/Mp_ATCC_25960.fastq -b  
gs://mybucket --label Mp_25960_b --organism "Mycoplasma pirum"
```

```
RAPT job has been created successfully.
```

```
-----  
Job-id:                5541b09bb9  
Output storage:        gs://mybucket/5541b09bb9  
GCP account:           111111111111-compute@developer.gserviceaccount.com  
GCP project:           example  
-----
```

```
[**Attention**] RAPT jobs may take hours to finish. Progress of this job can be  
viewed in GCP stackdriver log viewer at:
```

```
https://console.cloud.google.com/logs/viewer?project=strides-  
documentation-testing&filters=text:5541b09bb9
```

```
For current status of this job, run:
```

```
run_rapt_gcp.sh joblist | fgrep 5541b09bb9
```

```
For technical details of this job, run:
```

```
run_rapt_gcp.sh jobdetails 5541b09bb9
```

```
$
```

# GCP RAPT Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

## Checking progress

Execution information is available in the Google Console

Follow the link provided in the standard output to view the logs

Google Cloud Platform ncbi-rapt

CLASSIC CREATE METRIC CREATE SINK SHOW LIBRARY

text:7a2a132189

Global All logs Any log level Last hour

Jump to now

Showing logs from 4:51 PM to now (EDT) Download logs View Options

Load older logs

▶	i	2020-07-16 17:52:37.205 EDT	[2020-07-16 21:52:37] Usage metrics sent to NCBI	⋮
▶	*	2020-07-16 17:52:37.206 EDT	[2020-07-16 21:52:37] Pipeline starting ..	⋮
▶	*	2020-07-16 17:52:37.952 EDT	[2020-07-16 21:52:37] Found input: SRA run accession SRR9899747	⋮
▶	*	2020-07-16 17:52:38.608 EDT	[2020-07-16 21:52:38] Start downloading input-2020-06-10.build4646...	⋮
▶	i	2020-07-16 17:52:38.915 EDT	[2020-07-16 21:52:38] /misc_util/sra_check.sh->stdout: cloud retrie...	⋮
▶	i	2020-07-16 17:52:38.915 EDT	[2020-07-16 21:52:38] /misc_util/sra_check.sh->stderr: Read 5 spots...	⋮
▶	i	2020-07-16 17:52:41.068 EDT	[2020-07-16 21:52:41] Usage metrics sent to NCBI	⋮
▶	*	2020-07-16 17:52:41.077 EDT	[2020-07-16 21:52:41] Starting skesa ...	⋮
▶	i	2020-07-16 17:52:41.077 EDT	[2020-07-16 21:52:41] Running command skesa --contigs_out /gcloud-s...	⋮



## Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

# Check the status of all jobs executed in this project

### Execution information

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

```
$ ./run_rapt_gcp.sh joblist
GCP Account: [111111111111-compute@example.gserviceaccount.com]
Project: [example]
JOB_ID      USER      LABEL      SRR      STATUS  START_TIME  END_TIME  OUTPUT_URI
5541b09bb9   tester    SRR3496277  Running  2020-11-10T18:52:30
gs://mybucket/5541b09bb9
```

# GCP RAPT

## Start-to-finish

- Download the RAPT interface
- Explore RAPT
- Run RAPT
  - Starting with an SRA run
  - Starting with reads fasta
- Follow the progress
- Review the results

## Review the output files

Files are in the selected output gs bucket (i.e. gs://mybucket/5541b09bb9/output.tar.gz)

- View the output of the run in the Google storage bucket

```
$ gsutil ls gs://mybucket/5541b09bb9/  
output.tar.gz  
run.log
```

- Copy the output of the run in the Google storage bucket, and untar

```
$ gsutil cp -r gs://mybucket/5541b09bb9/ .  
$ cd 5541b09bb9  
$ tar -xzf output.tar.gz
```

Same files as for PGAP

+ skesa.out.fa: multifasta file of the assembly produced by SKESA

+ verbose.log

+ concise.log

# Coming next....

- RAPT and PGAP are on github

<https://github.com/ncbi/pgap>

<https://github.com/ncbi/rapt>

- Subscribe and stay informed of new releases
- Soon....
  - Performance improvements in PGAP and RAPT (Dec 2020)
  - A web service for RAPT (Jan 2021)



# We want your feedback !

- Will these tools be useful to your research?
- What other tools would help you reach your goals?
- How do you use annotated genomes?
- What computing environment do you have access to?
  
- Are you interested in getting sneak previews of new tools or features, and in becoming a tester?
  
- Talk to us during this workshop or later,
- [prokaryote-tools@ncbi.nlm.nih.gov](mailto:prokaryote-tools@ncbi.nlm.nih.gov)
- [thibaudf@nih.gov](mailto:thibaudf@nih.gov)



# Acknowledgements

David Arndt

Azat Badretdin

Slava Chetvernin

Rob Cohen

Wratko Hlavina

Wenjun Li

Shennan Lu

Peter Meric

Eyal Mozes

Douglas Slotta

Daniel Soren

Deacon Sweeney

Lukas Wagner

Mingzhang Yang

Ben Busby

Lewis Geer

Jim Ostell

Kim Pruitt

Eugene Yaschenko

Steve Turner

This research was supported by the Intramural Research Program of the NIH, National Library of Medicine



U.S. National Library of Medicine  
National Center for Biotechnology Information

