PGTB: Bioinformatics Workshop

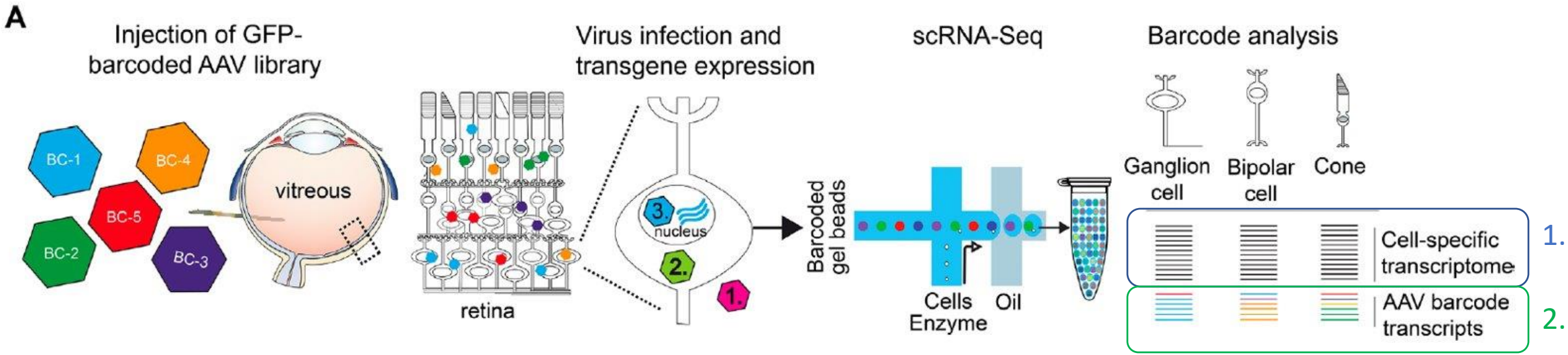
Molly Johnson

Outline

- Morning session
  - Quick scAAVengr overview/refresher
  - ScAAVengr pipeline
    - Setting up
      - HPC
      - Downloading data/tools
      - Git repositories
      - Conda env
    - Running pre-processing
- Lunch Break
- Afternoon session
  - Running analysis
    - Single cell tutorial
      - Scanpy
      - Other QC tools/best practices
    - Plotting

scAAVengr

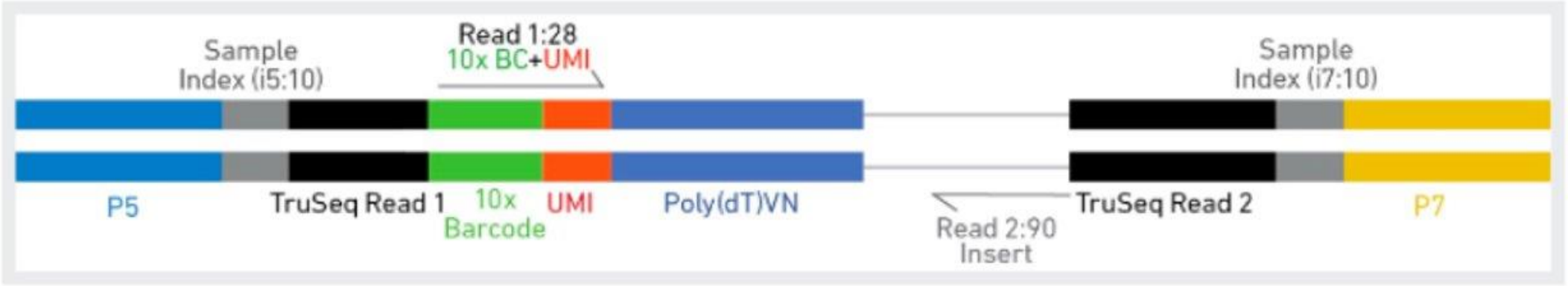
Identify AAV capsid variants with GFP-barcoded transgene



1. scRNAseq
2. GFP quantification

scRNAseq → scAAvengr

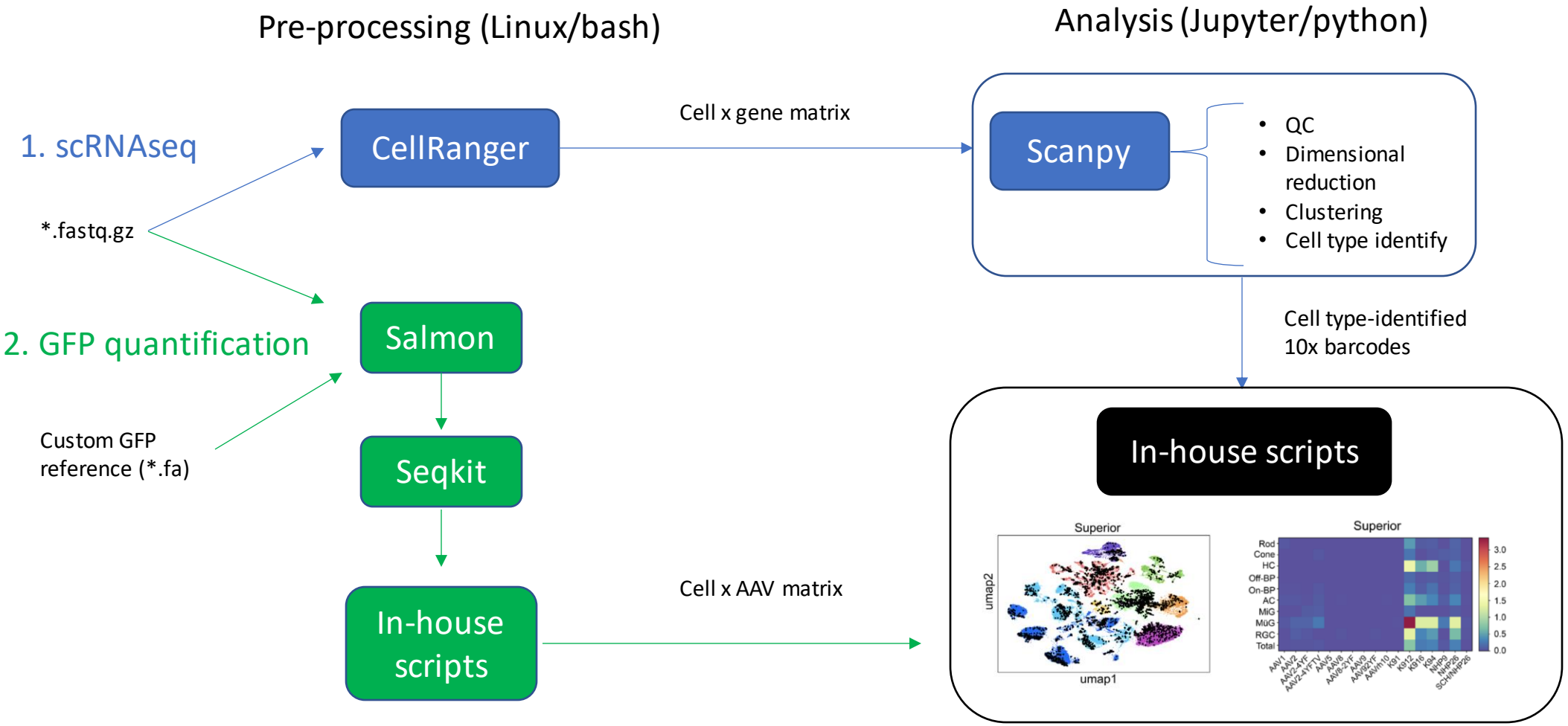
- 10x scRNAseq
  - Read 1: Captures unique cell barcode and unique molecular identifier
    - *Allows us to map to a particular cell/cell type and quantify unique mRNA molecules for expression analysis*
  - Read 2: Captures mRNA via polyA tail
    - *Our GFP reads will be sequenced towards the 3' end of the read, which is where our GFP barcode is located*
    - *We sequence R2 at 151bp (instead of default 92bp) to ensure that we capture our GFP barcodes*
    - *We can then use this GFP barcode to map to the AAV*



What is a computational pipeline?

- A collection of scripts, tools, and processes used to collect raw data from multiple sources, analyze it, and present results
  - Script
    - File containing commands in a particular language (i.e. pyton, bash, etc.) to execute processes
  - Tool
    - Published/executable computational software
    - Ex: 10x CellRanger
      - *Side note: also comprised of their own scripts

scAAVengr pipeline

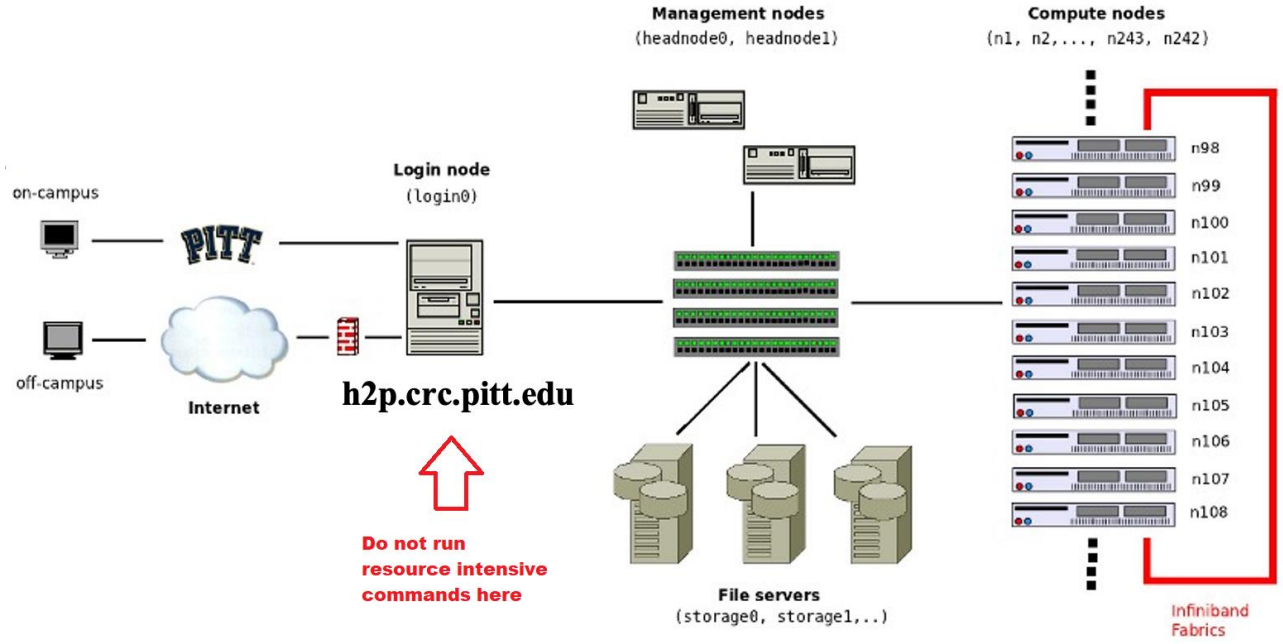


Setting up for pipeline

1. HPC environment
   - High-performance computing environment with more CPUs and memory to analyze large datasets
   - Can connect remotely to large computational infrastructure to run pipeline
   - We will be using Pitt HTC CRC primarily
     - If you do not have Pitt access, please use CMU HPC or PSC (or partner up)
2. Datasets
   - scRNAseq samples
   - References
     - Human genome
     - GFP construct
3. Git repository
   - Sharing code/scripts
   - Clone repository on Pitt HTC CRC to run pipeline
4. Conda
   - Manages software to use within scAAVengr pipeline
   - Ensures everyone is using correct version of each bioinformatics tool

HPC - Pitt HTC CRC

- High-performance computing system
  - Access remotely/securely via ssh and command-line interface



https://crc.pitt.edu/user-manual/job-scheduling-policy

HPC - Pitt HTC CRC

- Connecting remotely
  - VPN
  - SSH
- Interacting with resources
  - Bash
    - Language used to move around file system and interact with files
  - Emacs
    - Opening script/file to edit and save
  - Slurm workload manager
    - Submitting scripts/computer programs to a 'node' to run

Accessing Pitt HTC CRC

- Pitt requires a secure VPN connection to access remote HPC system
  - Requires 2FA
  - Install instructions:
    - https://www.technology.pitt.edu/services/pittnet-vpn-globalprotect
    - Login to my.pitt.edu
    - Software download page: https://software.pitt.edu/
- On your laptop, search/open up ‘Pulse Secure’ or 'GlobalProtect'
  - Enter username and password, then PUSH
  - 2-factor authentication required

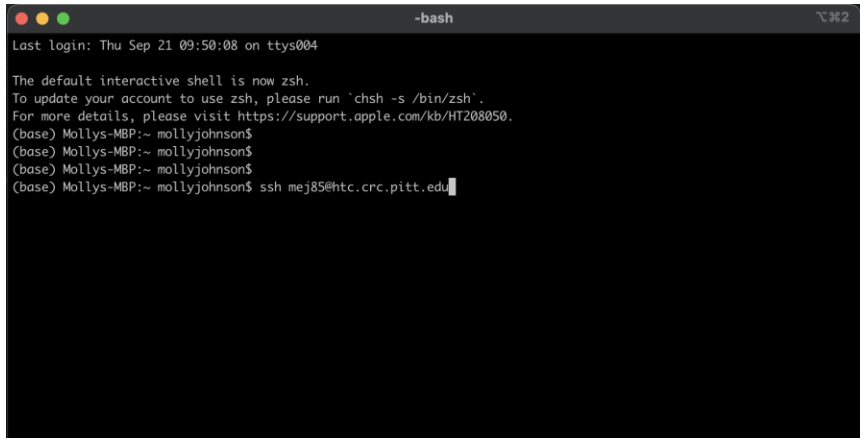
Interacting with Pitt HTC CRC

- Terminal is a command-line interface for your computer
  - Allows you to interact/navigate your computer using code
  - We will use this to connect to HTC CRC
  - https://crc.pitt.edu/getting-started/accessing-cluster
    - See more info and **Windows-based** instructions here

- On your laptop, search/open application 'Terminal'
  - If windows, open up 'PuTTY'

- Log in to HTC CRC

  ssh <pitt_username>@htc.crc.pitt.edu



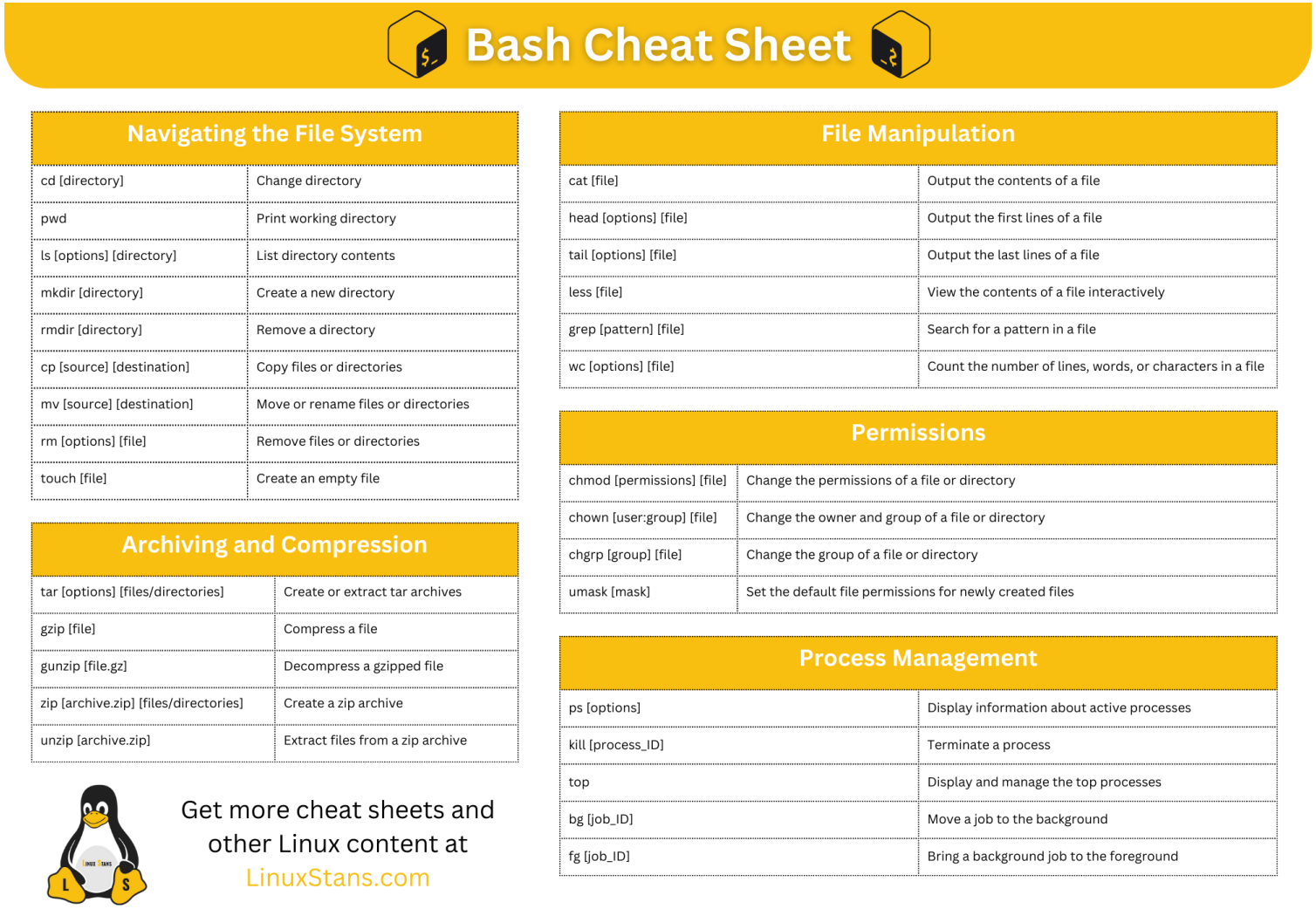
'ssh' is a command used to establish a secure remote connection to another computer

You will have a Pitt account set-up (temporarily) on HTC CRC

Address/name of computer that you are connecting to

Interacting with Pitt HTC CRC:
Quick Bash Reference Guide

- Default linux-based programming language to move around / interact with computer
- Useful commands:
  - cd <source> <dest>
    - 'Change Directory' from **source** to **destination**
  - ls <dir>
    - 'List' contents in your directory
  - pwd
    - 'Print Working Directory'
  - cat <file>
    - Output file contents to screen
  - head <file>
    - Print first 5 lines of file
  - *Be EXTREMELY cautious with rm (remove); you can't undo it



Bash Cheat Sheet

Navigating the File System

cd [directory]	Change directory
pwd	Print working directory
ls [options] [directory]	List directory contents
mkdir [directory]	Create a new directory
rmdir [directory]	Remove a directory
cp [source] [destination]	Copy files or directories
mv [source] [destination]	Move or rename files or directories
rm [options] [file]	Remove files or directories
touch [file]	Create an empty file

Archiving and Compression

tar [options] [files/directories]	Create or extract tar archives
gzip [file]	Compress a file
gunzip [file.gz]	Decompress a gzipped file
zip [archive.zip] [files/directories]	Create a zip archive
unzip [archive.zip]	Extract files from a zip archive

Get more cheat sheets and other Linux content at LinuxStans.com

File Manipulation

cat [file]	Output the contents of a file
head [options] [file]	Output the first lines of a file
tail [options] [file]	Output the last lines of a file
less [file]	View the contents of a file interactively
grep [pattern] [file]	Search for a pattern in a file
wc [options] [file]	Count the number of lines, words, or characters in a file

Permissions

chmod [permissions] [file]	Change the permissions of a file or directory
chown [user:group] [file]	Change the owner and group of a file or directory
chgrp [group] [file]	Change the group of a file or directory
umask [mask]	Set the default file permissions for newly created files

Process Management

ps [options]	Display information about active processes
kill [process_ID]	Terminate a process
top	Display and manage the top processes
bg [job_ID]	Move a job to the background
fg [job_ID]	Bring a background job to the foreground

Let's try it out

- We're going to make a project directory in our home directory for our scAAVengr output
  - Log in to Pitt HTC CRC:

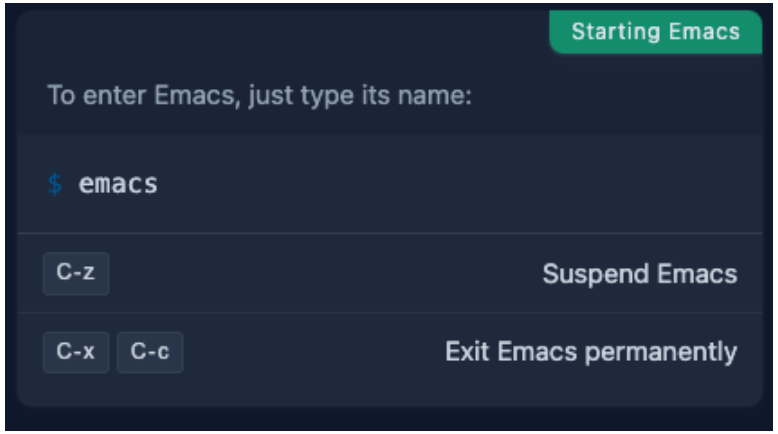
```
Mollys-MacBook-Pro:~ mollyjohnson$ ssh mej85@htc.crc.pitt.edu
mej85@htc.crc.pitt.edu's password:
```

  - Using bash, create directory and move around directory structure

```
[mej85@login0b ~]$ mkdir ~/pgtb_human_retina_explant
[mej85@login0b ~]$
[mej85@login0b ~]$ cd ~/pgtb_human_retina_explant/
[mej85@login0b pgtb_human_retina_explant]$
[mej85@login0b pgtb_human_retina_explant]$ mkdir OUT_ERR
[mej85@login0b pgtb_human_retina_explant]$
[mej85@login0b pgtb_human_retina_explant]$ ls
OUT_ERR
[mej85@login0b pgtb_human_retina_explant]$ pwd
/ihome/lbyrne/mej85/pgtb_human_retina_explant
[mej85@login0b pgtb_human_retina_explant]$
[mej85@login0b pgtb_human_retina_explant]$
```

Interacting with Pitt HTC CRC:
Emacs for file editing

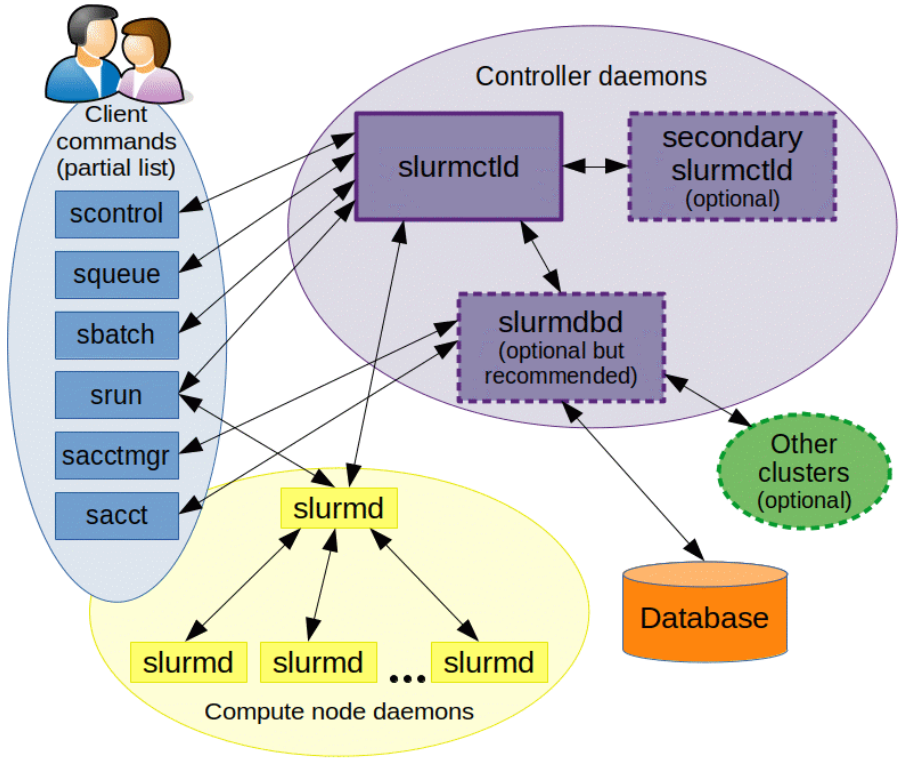
- Text editor for POSIX operating systems (Linux, Mac, Windows, etc.)
  - Many commands, but a few that we'll be using:
    - Emacs
      - Open emacs editor interactively on command line
    - Use arrow keys to move around and add any text
    - C-x, C-s
      - Save edits/changes
    - C-x, C-c
      - Exit emacs after saving changes
  - We will use this later on when we try creating our own script



https://quickref.me/emacs.html

Interacting with Pitt HTC CRC:
Slurm Workload Manager

- Submits/manages jobs on cluster
- Common commands
  - **sbatch <script>**
    - Submits 'job'
      - A Job is just a term we use for a computer program/script that's running on the cluster
  - **squeue –u <userid>**
    - Tracks submitted/running jobs
  - **sacct –j <jobid>**
    - Looks at status of completed job
      - Whether it completed successfully/failed, time it took to run, etc.



https://slurm.schedmd.com/quickstart.html

Datasets

- If working on Pitt HTC CRC:
  - Samples are under /bgfs/lbyrne/PGTB/data/
  - Reference is under /bgfs/lbyrne/PGTB/resources/
- If working on CMU HPC or PSC:
  - Download datasets on command-line:
    - Create OAuth 2.0 access token:
      - Info: https://cloud.google.com/storage/docs/authentication#apiauth
      - https://developers.google.com/oauthplayground/
        - Cloud Storage API V1
          - Full_control
          - Read_only
          - Write_only
        - Authorize APIs
        - Exchange auhtorization code for tokens; copy ACCESS token
    - Download with curl / JSON
      - https://cloud.google.com/storage/docs/downloading-objects#download-object-json
      - Ex (downloading yml conda env):
        - curl -X GET -H "Authorization: Bearer <ACCESS_TOKEN>" -o "/path/to/scaavengr_env.yml" https://www.googleapis.com/storage/v1/b/2023_pgtb/o/resources%2Fscaavengr_env.yml?alt=media
  - Local Download
    - https://console.cloud.google.com/storage/browser/2023_pgtb

Let's check out the datasets

```
[mej85@login0b ~]$ cd /bgfs/lbyrne/PGTB/
[mej85@login0b PGTB]$ ls
data  metadata  resources
[mej85@login0b PGTB]$ ls data/
LB1_BYR819A1_S1_R1_001.fastq.gz  LB1_BYR819A1_S1_R2_001.fastq.gz  LB4_BYR819A4_S4_R1_001.fastq.gz  LB4_BYR819A4_S4_R2_001.fastq.gz
[mej85@login0b PGTB]$ zcat data/LB1_BYR819A1_S1_R1_001.fastq.gz | head
@A00523:114:H3J57DRXY:1:1101:1289:1000 1:N:0:GTAACATGCG+NGGTAACACT
NATAGACCAGTCGGTCTGTGGCTTCATA
+
#FFF:FFF:FFFFFFFFFFFFF:FFFF
@A00523:114:H3J57DRXY:1:1101:1307:1000 1:N:0:GTAACATGCG+NGGTAACACT
NGAGCGCGTAACAGGCCACTAGCTCGAC
+
#FFFFFFFFFFFFFFFFFFFFFFFFFF
@A00523:114:H3J57DRXY:1:1101:1651:1000 1:N:0:GTAACATGCG+NGGTAACACT
NGGGAAGCAAACTAGAATTCCTTAGCCT
[mej85@login0b PGTB]$ ls resources/
Anaconda3-2020.11-Linux-x86_64.sh  gfp_barcodes_25bp.fa  gfp_barcodes.fa  gfp_barcodes_salmon_index  gfpbc_2_aav_key.csv  refdata-gex-GRCh38-2020-A
[mej85@login0b PGTB]$ head  resources/gfp_barcodes_25bp.fa
>Barcode1
CTGTAGTCTACACGACATCATCTAG
>Barcode2
ATCGTCTACAGTACGAGCGCTCTAC
>Barcode3
ACAGACTGTCTACGACGACTATAGT
>Barcode4
GTACTCGCATGTGATCGCAGTGCAG
>Barcode5
CTGCGCATATCGTACATGTGTCTGA
[mej85@login0b PGTB]$
```

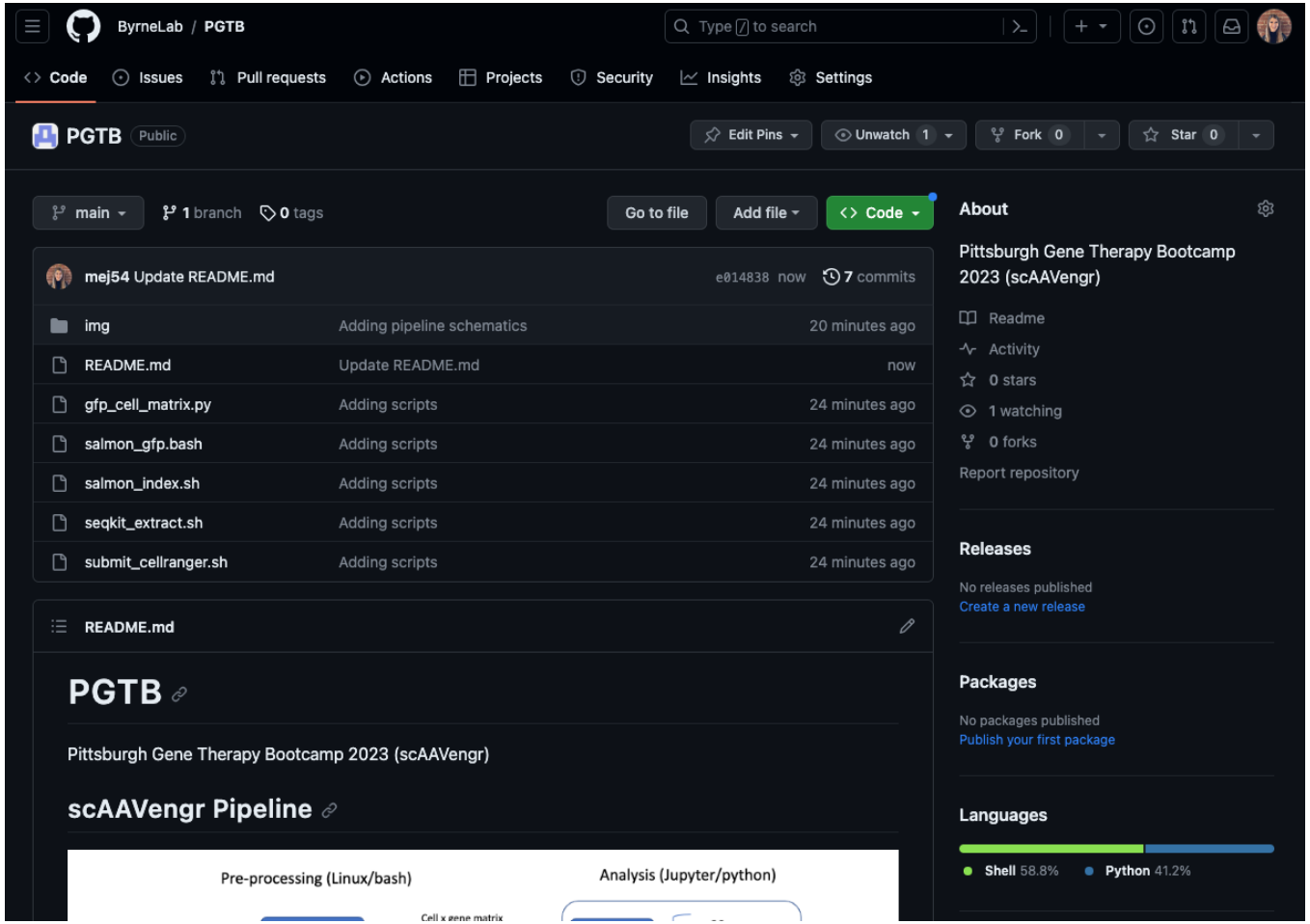
GitHub

- Location for sharing scripts/pipeline/tools
  - Tracks changes
- Repositories
  - A group of scripts/files for a particular computational tool, pipeline, or project
- https://github.com/ByrneLab/PGTB

On Pitt HTC CRC:

- mkdir ~/git_repos
- cd ~/git_repos
- git clone https://github.com/ByrneLab/PGTB.git

*You now have the scAAVengr pipeline in your directory on Pitt CRC under ~/git_repos/PGTB



What is a computational pipeline?

- A collection of scripts, tools, and processes used to collect raw data from multiple sources, analyze it, and present results
  - **Script**
    - **File containing commands in a particular language (i.e. pyton, bash, etc.) to execute processes**
  - Tool
    - Published/executable computational software
    - Ex: 10x CellRanger
      - *Side note: also comprised of their own scripts

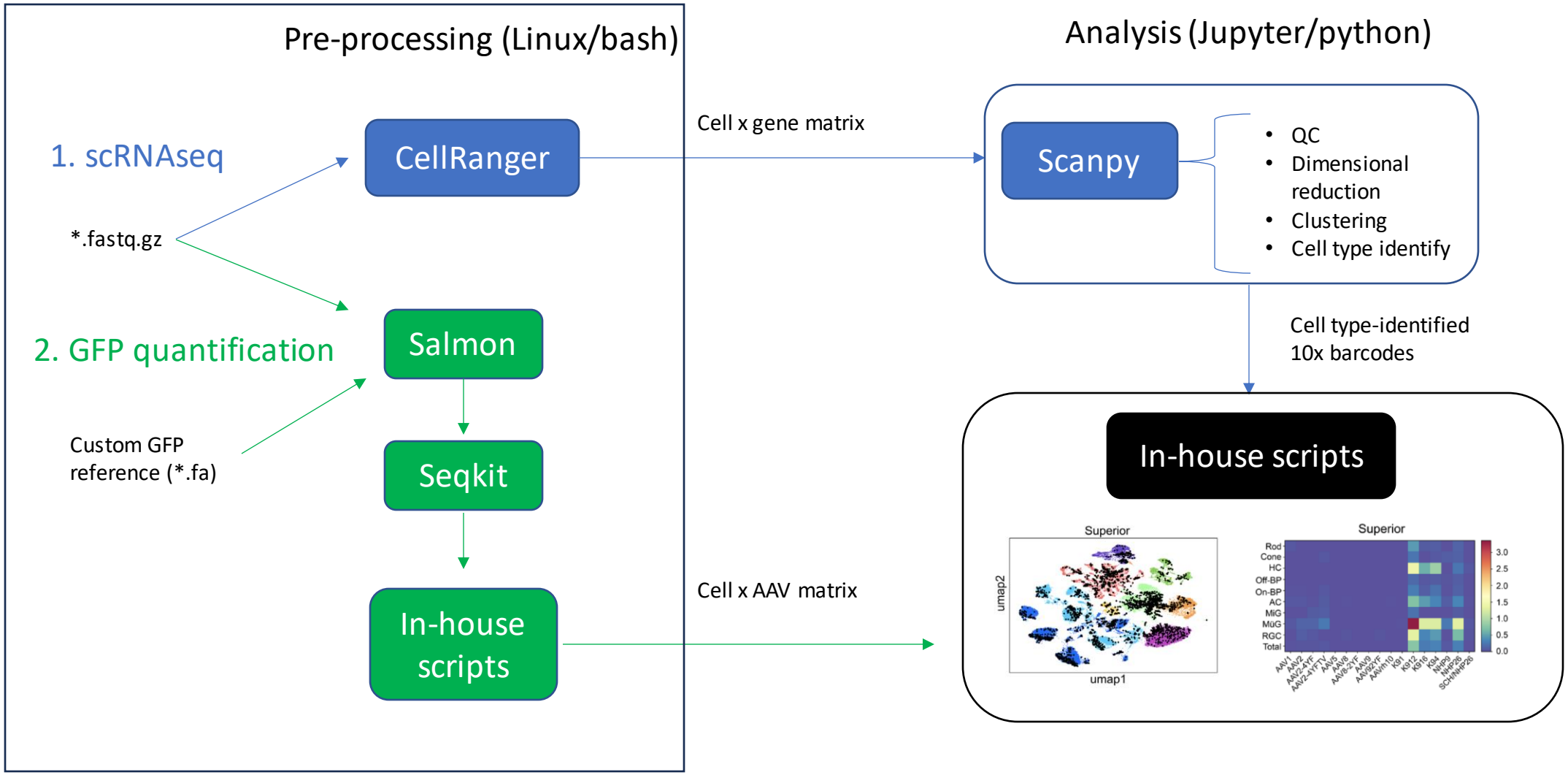
What is in a script?

- Interpreter
  - Tells computer which language to use to execute script
- Environment
  - Computational tools loaded into the script that can then be accessed/used in that script
- Arguments/flags
  - Parameters or information you send to the script that modifies behavior of script
- Variables
  - Contain or point to objects (text, number, data structures, etc)
- Commands
  - Executing processes

*We will practice creating a script after we get the pipeline running

```
#!/usr/bin/env python

import argparse
import array
import numpy as np
import pandas as pd
import sys

################ MAIN

# GET OPTIONS
parser = argparse.ArgumentParser()
parser.add_argument('--bc_uniq')
parser.add_argument('--gfpbc')
parser.add_argument('--out')
parser.add_argument('--outbin')
args = parser.parse_args()

barcode_10x_txt = args.bc_uniq # readname to 10x bc to 10x umi
gfpbc_file = args.gfpbc # readnames to GFP barcode file
out_file = args.out
out_file_binary = args.outbin

file1 = open(gfpbc_file,"r")
gfpbc_results = file1.read().splitlines()
file1.close()


info_10x = pd.read_csv(barcode_10x_txt,header=None,sep="\t")
c_10x_barcode = dict(zip(info_10x[0],info_10x[1]))
c_umi = dict(zip(info_10x[0],info_10x[2]))
c_10xbarcode_2_virus = {}
gfp_bc_all = []
count=0
#full_10x_bc_umi_with_gfp = []
```

scAAVengr pipeline



Pre-run checks

- Check for CellRanger human reference genome
  - Downloaded from 10x: https://www.10xgenomics.com/support/software/cell-ranger/downloads

```
[mej85@login0b PGTB]$ ls /bgfs/lbyrne/PGTB/resources/refdata-gex-GRCh38-2020-A
fasta  genes  pickle  reference.json  star
[mej85@login0b PGTB]$
[mej85@login0b PGTB]$ ls /bgfs/lbyrne/PGTB/resources/refdata-gex-GRCh38-2020-A/fasta/
genome.fa  genome.fa.fai
[mej85@login0b PGTB]$ head -5 /bgfs/lbyrne/PGTB/resources/refdata-gex-GRCh38-2020-A/fasta/genome.fa
>chr1 1
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
[mej85@login0b PGTB]$
[mej85@login0b PGTB]$ head -10 /bgfs/lbyrne/PGTB/resources/refdata-gex-GRCh38-2020-A/genes/genes.gtf
##description: evidence-based annotation of the human genome (GRCh38), version 32 (Ensembl 98)
##provider: GENCODE
##contact: gencode-help@ebi.ac.uk
##format: gtf
##date: 2019-09-05
chr1    HAVANA  gene    29554   31109   .       +       .       gene_id "ENSG00000243485"; gene_version "5"; gene_type "lncRNA"; gene_name "MIR1302-2HG"; level 2;
 "ncRNA_host"; havana_gene "OTTHUMG00000000959.2";
chr1    HAVANA  transcript      29554   31097   .       +       .       gene_id "ENSG00000243485"; gene_version "5"; transcript_id "ENST00000473358"; transcript_ve
RNA"; gene_name "MIR1302-2HG"; transcript_type "lncRNA"; transcript_name "MIR1302-2HG-202"; level 2; transcript_support_level "5"; hgnc_id "HGNC:52482"; tag "not_b
tag "dotter_confirmed"; tag "basic"; havana_gene "OTTHUMG00000000959.2"; havana_transcript "OTTHUMT00000002840.1";
chr1    HAVANA  exon    29554   30039   .       +       .       gene_id "ENSG00000243485"; gene_version "5"; transcript_id "ENST00000473358"; transcript_version "1
ne_name "MIR1302-2HG"; transcript_type "lncRNA"; transcript_name "MIR1302-2HG-202"; exon_number 1; exon_id "ENSE00001947070"; exon_version "1"; level 2; transcript
id "HGNC:52482"; tag "not_best_in_genome_evidence"; tag "dotter_confirmed"; tag "basic"; havana_gene "OTTHUMG00000000959.2"; havana_transcript "OTTHUMT00000002840.
chr1    HAVANA  exon    30564   30667   .       +       .       gene_id "ENSG00000243485"; gene_version "5"; transcript_id "ENST00000473358"; transcript_version "1
ne_name "MIR1302-2HG"; transcript_type "lncRNA"; transcript_name "MIR1302-2HG-202"; exon_number 2; exon_id "ENSE00001922571"; exon_version "1"; level 2; transcript
id "HGNC:52482"; tag "not_best_in_genome_evidence"; tag "dotter_confirmed"; tag "basic"; havana_gene "OTTHUMG00000000959.2"; havana_transcript "OTTHUMT00000002840.
chr1    HAVANA  exon    30976   31097   .       +       .       gene_id "ENSG00000243485"; gene_version "5"; transcript_id "ENST00000473358"; transcript_version "1
ne_name "MIR1302-2HG"; transcript_type "lncRNA"; transcript_name "MIR1302-2HG-202"; exon_number 3; exon_id "ENSE00001827679"; exon_version "1"; level 2; transcript
id "HGNC:52482"; tag "not_best_in_genome_evidence"; tag "dotter_confirmed"; tag "basic"; havana_gene "OTTHUMG00000000959.2"; havana_transcript "OTTHUMT00000002840.
[mej85@login0b PGTB]$
```

Pre-run checks

- Check/create salmon GFP reference file

```
[mej85@login0b PGTB]$ ls /bgfs/lbyrne/PGTB/resources/
Anaconda3-2020.11-Linux-x86_64.sh  gfp_barcodes_25bp.fa  gfp_barcodes.fa  gfp_barcodes_salmon_index  gfpbc_2_aav_key.csv  refdata-gex-GRCh38-2020-A
[mej85@login0b PGTB]$
[mej85@login0b PGTB]$ head -2 /bgfs/lbyrne/PGTB/resources/gfp_barcodes.fa
>Barcode1
ATGGTGAGCAAGGGCGAGGAGCTGTTCACCGGGGTGGTGCCCATCCTGGTCGAGCTGGACGGCGACGTAAACGGCCACAAGTTCAGCGTGTCCGGCGAGGGCGAGGGCGATGCCACCTACGGCAAGCTGACCCTGAAGTTCATCTGCAC
CACCCTGACCTACGGCGTGCAGTGCTTCAGCCGCTACCCCGACCACATGAAGCAGCACGACTTCTTCAAGTCCGCCATGCCCGAAGGCTACGTCCAGGAGCGCACCATCTTCTTCAAGGACGACGGCAACTACAAGACCCGCGCCGAGG
TGAAGGGCATCGACTTCAAGGAGGACGGCAACATCCTGGGGCACAAGCTGGAGTACAACTACAACAGCCACAACGTCTATATCATGGCCGACAAGCAGAAGAACGGCATCAAGGTGAACTTCAAGATCCGCCACAACATCGAGGACGGC
ATCGGCGACGGCCCCGTGCTGCTGCCCGACAACCACTACCTGAGCACCCAGTCCGCCCTGAGCAAAGACCCCAACGAGAAGCGCGATCACATGGTCCTGCTGGAGTTCGTGACCGCCGCCGGGATCACTCTCGGCATGGACGAGCTGTA
TCTACACGACATCATCTAGTCGACTAGAGCTCGCTGATCAGCCTCGACTGTGCCTTCTAGTTGCCAGCATCTGTTGTTTGCCCCTCCCCCGTGCCTTCCTTGACCCTGGAAGGTGCCACTCCCACTGTCCTTTCCTAATAAAATGAGGA
GGGGGGGGGGTGGGGCAGGACAGCAAGGGGGAGGATTGGGAAGACAATAGCAG
[mej85@login0b PGTB]$
[mej85@login0b PGTB]$ ls /bgfs/lbyrne/PGTB/resources/gfp_barcodes_salmon_index
complete_ref_lens.bin  ctg_offsets.bin        info.json  pos.bin          rank.bin            ref_indexing.log  refseq.bin  versionInfo.json
ctable.bin             duplicate_clusters.tsv  mphf.bin   pre_indexing.log  refAccumLengths.bin  reflengths.bin    seq.bin
[mej85@login0b PGTB]$
[mej85@login0b PGTB]$ head /bgfs/lbyrne/PGTB/resources/gfp_barcodes_salmon_index/info.json
{
    "index_version": 4,
    "reference_gfa": [
        "/bgfs/lbyrne/PGTB/resources/gfp_barcodes_salmon_index"
    ],
    "sampling_type": "dense",
    "k": 31,
    "num_kmers": 2194,
    "num_contigs": 60,
    "seq_length": 3994,
[mej85@login0b PGTB]$
```

Running Pre-Processing Pipeline

1. Cellranger
   - Create [gene x cell] matrix
2. Salmon
   - Quantify GFP reads
3. Seqkit
   - Extract GFP barcodes
4. In-house script
   - Create [gfp barcode x cell] matrix

Running Pre-Processing Pipeline

1. **Cellranger**
   - **Create [gene x cell] matrix**
2. Salmon
   - Quantify GFP reads
3. Seqkit
   - Extract GFP barcodes
4. In-house script
   - Create [gfp barcode x cell] matrix

Submit CellRanger

Executing script

```
sbatch -o `pwd`/OUT_ERR/cellranger_%j.out ~/git_repos/PGTB/submit_cellranger.sh \
    -p `pwd` -d /bgfs/lbyrne/PGTB/data/ \
    -n LB1_BYR819A1 \
    -r /bgfs/lbyrne/PGTB/resources/refdata-gex-GRCh38-2020-A/
```

Code within the script itself

```
#!/bin/bash

#SBATCH --cluster htc
#SBATCH --cpus-per-task=32
#SBATCH -t 23:00:00

usage() { echo -e "SCRIPT FOR CELLRANGER COUNT\nUsage: sbatch -o <PROJ_DIR>/OUT_ERR/cellranger_%j.out $0 [-p </path/to/projectdir>]
 [-d </path/to/fastqdir>] [-n <sample_name>] [-r </path/to/cellranger_ref>]" 1>&2; exit 1; }

while getopts ":p:d:n:r:" arg; do
    case $arg in
        p)
            PROJ_DIR=${OPTARG}
            ;;
        d)
            FASTQ_DIR=${OPTARG}
            ;;
```

Running Pre-Processing Pipeline

1. Cellranger
   - Create [gene x cell] matrix
2. **Salmon**
   - **Quantify GFP reads**
3. Seqkit
   - Extract GFP barcodes
4. In-house script
   - Create [gfp barcode x cell] matrix

Submit Salmon

Executing script

```
sbatch -o `pwd`/OUT_ERR/salmon_gfp_quant_%j.out  ~/git_repos/PGTB/salmon_gfp.sh \
    -p `pwd` -d /bgfs/lbyrne/PGTB/data/ \
    -n LB1_BYR819A1_S1 \
    -r /bgfs/lbyrne/PGTB/resources/gfp_barcodes_salmon_index/
```

Code within the script itself

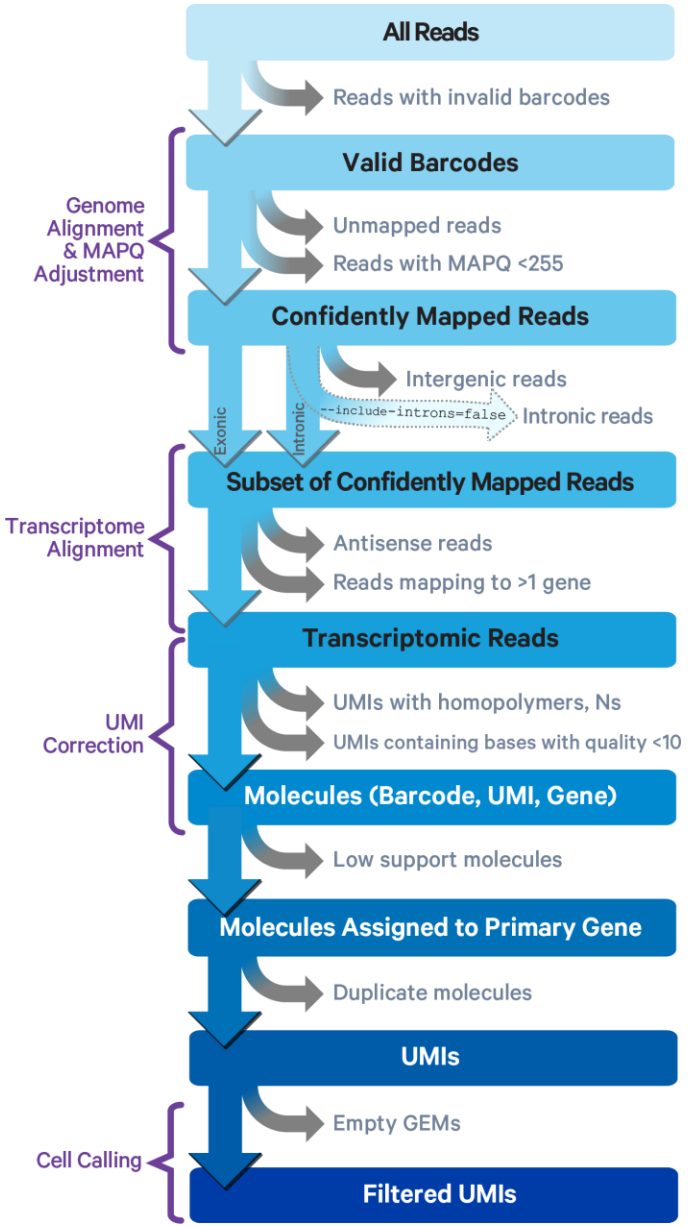
```
#!/bin/bash

#SBATCH --cluster htc
#SBATCH --cpus-per-task=8
#SBATCH -t 2:00:00 # Runtime in D-HH:MM


usage() { echo -e "SCRIPT FOR SALMON GFP QUANTIFICATION\nUsage: sbatch -o <PROJ_DIR>/OUT_ERR/salmon_gfp_quant_%j.out $0 [-p </path/\
to/projectdir>] [-d </path/to/fastqdir>] [-n <sample_name>] [-r </path/to/salmon_index_ref>]" 1>&2; exit 1; }

while getopts ":p:d:n:r:" arg; do
    case $arg in
        p)
            PROJ_DIR=${OPTARG}
            ;;
        d)
            FASTQ_DIR=${OPTARG}
            ;;
```

Pre-processing Pipeline:
Cellranger count

- R1
  - Contains 10x barcode (16 bp)
    - Unique identifier for the droplet that encapsulates 1 cell (in theory, a unique cell identifier)
      - Can be affected by empty droplets or droplets that contain more than 1 cell
  - Contains UMI (12 bp)
    - Unique molecular identifier
      - Each unique mRNA molecule has a unique sequence associated with it
      - Allows for de-duplication of PCR effects to allow for more accurate expression quantification
- R2
  - Contains 3' polya mRNA
    - Aligned to reference genome to identify which gene this mRNA originates from

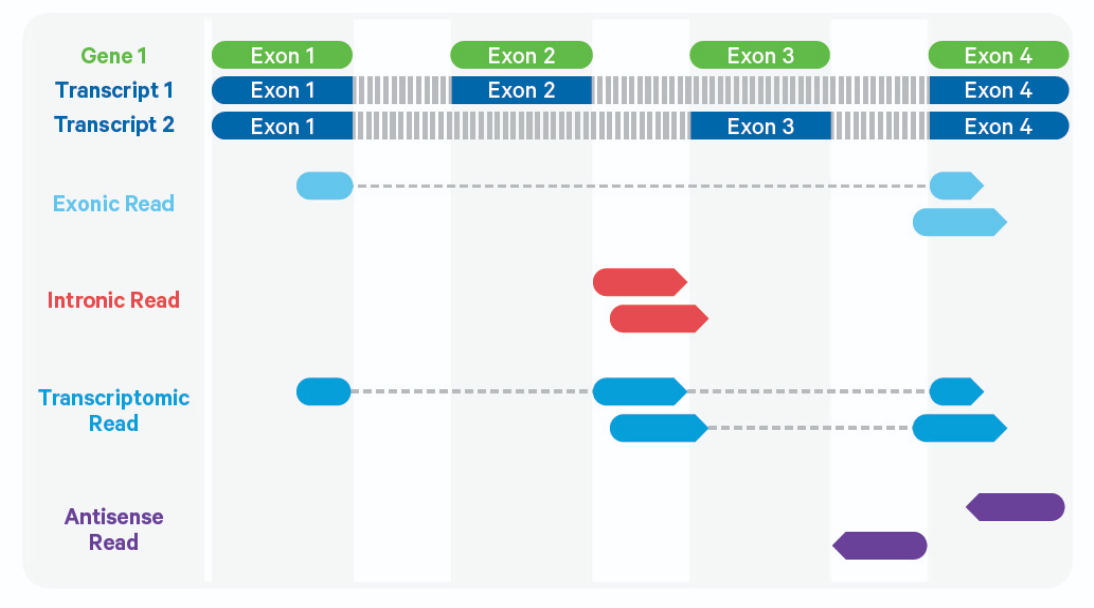
Pre-processing Pipeline:
Cellranger count

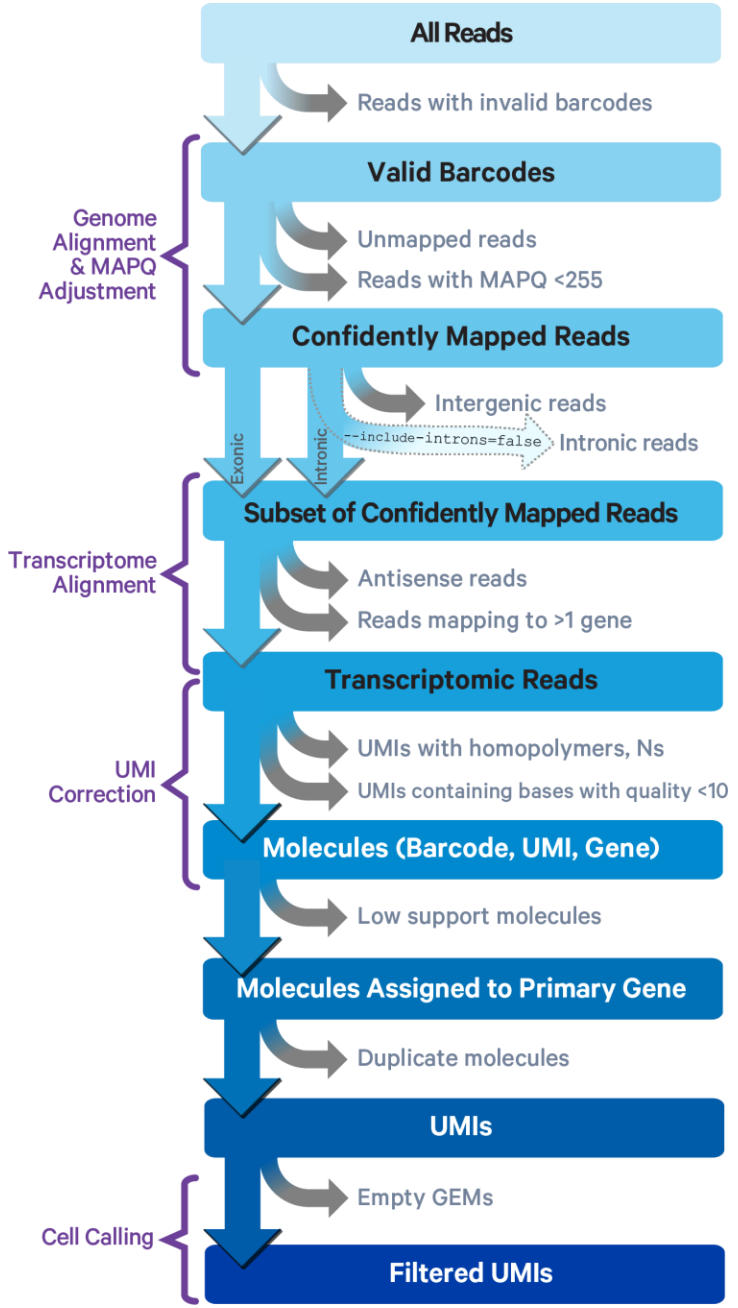
- R1
  - Contains 10x barcode (16 bp)
    - Unique identifier for the droplet that encapsulates 1 cell (in theory, a unique cell identifier)
      - Can be affected by empty droplets or droplets that contain more than 1 cell
  - Contains UMI (12 bp)
    - Unique molecular identifier
      - Each unique mRNA molecule has a unique sequence associated with it
      - Allows for de-duplication of PCR effects to allow for more accurate expression quantification
- R2
  - Contains 3' polya mRNA
    - Aligned to reference genome to identify which gene this mRNA originates from



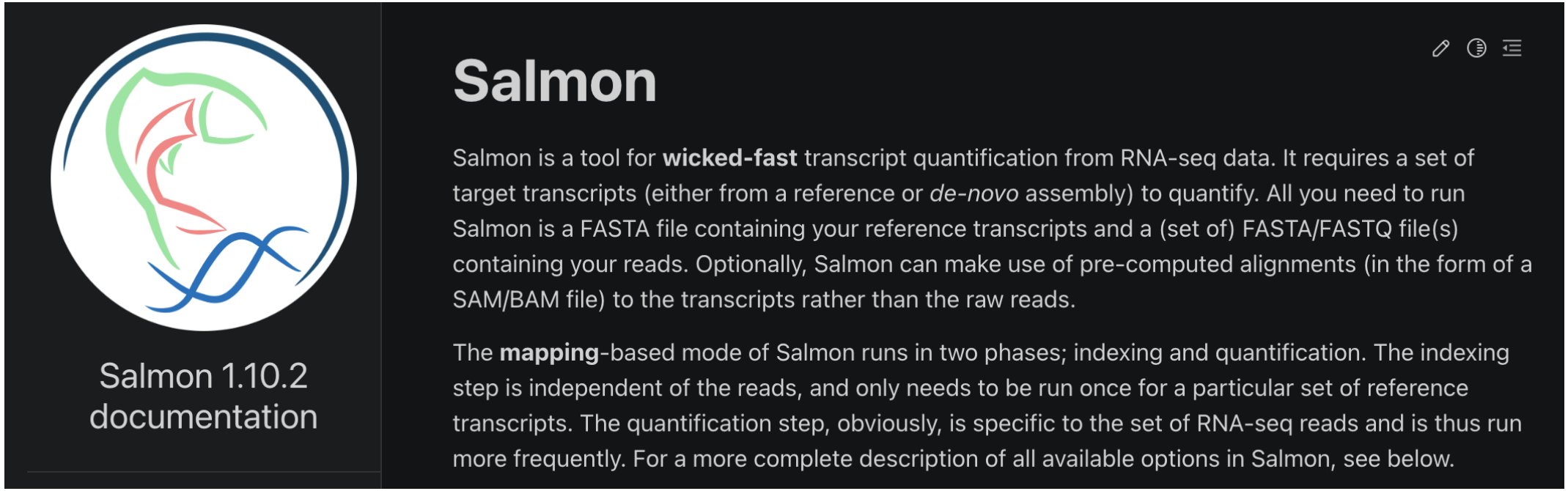
Pre-processing Pipeline:
Cellranger count

- Genome alignment
  - Uses STAR aligner – performs splicing-aware alignment to genome
  - FA file
    - Reference file with chromosomes and genomic sequence
  - GTF file
    - Annotation file used in conjunction with reference file
    - Annotates the genes in the genome and the coordinates
    - Reads classified as 'exonic', 'intronic', or 'intergenic'





Salmon – GFP quantification



https://salmon.readthedocs.io/en/latest/salmon.html

Running Pre-Processing Pipeline

1. Cellranger
   - Create [gene x cell] matrix
2. Salmon
   - Quantify GFP reads
3. **Seqkit**
   - **Extract GFP barcodes**
4. In-house script
   - Create [gfp barcode x cell] matrix

Submit Seqkit

Executing script

```
sbatch -o `pwd`/OUT_ERR/seqkit_locate_%j.out ~/git_repos/PGTB/seqkit_extract.sh \
    -p `pwd` -d /bgfs/lbyrne/PGTB/data/ \
    -n LB1_BYR819A1_S1 \
    -b /bgfs/lbyrne/PGTB/resources/gfp_barcodes_25bp.fa
```

Code within the script itself

```
#!/bin/bash

#SBATCH --cluster htc
#SBATCH --cpus-per-task=2
#SBATCH -t 2:00:00 # Runtime in D-HH:MM

usage() { echo -e "SCRIPT FOR SEQKIT GFP READ EXTRACT\nUsage: sbatch -o <PROJ_DIR>/OUT_ERR/seqkit_%j.out $0 [-p </path/to/projectdi
r>] [-d </path/to/fastqdir>] [-n <sample_name>] [-b <gfp_barcodes_fasta>]" 1>&2; exit 1; }

while getopts ":p:d:n:b:" arg; do
    case $arg in
        p)
            PROJ_DIR=${OPTARG}
            ;;
        d)
            FASTQ_DIR=${OPTARG}
            ;;
```

Seqkit – extract GFP barcodes

locate

Usage

```
locate subsequences/motifs, mismatch allowed

Attentions:

  1. Motifs could be EITHER plain sequence containing "ACTGN" OR regular
     expression like "A[TU]G(?:.{3})+?[TU](?:AG|AA|GA)" for ORFs.
  2. Degenerate bases/residues like "RYMM.." are also supported by flag -d.
     But do not use degenerate bases/residues in regular expression, you need
     convert them to regular expression, e.g., change "N" or "X"  to ".".
  3. When providing search patterns (motifs) via flag '-p',
     please use double quotation marks for patterns containing comma,
     e.g., -p '"A{2,}"' or -p "\"A{2,}\"". Because the command line argument
     parser accepts comma-separated-values (CSV) for multiple values (motifs).
     Patterns in file do not follow this rule.
  4. Mismatch is allowed using flag "-m/--max-mismatch",
     you can increase the value of "-j/--threads" to accelerate processing.
  5. When using flag --circular, end position of matched subsequence that
     crossing genome sequence end would be greater than sequence length.

Usage:
  seqkit locate [flags]

Flags:
      --bed                     output in BED6 format
  -c, --circular                circular genome. type "seqkit locate -h" for details
  -d, --degenerate              pattern/motif contains degenerate base
```

stats
sum
faidx
watch
sana
scat
fq2fa
fa2fq
fx2tab & tab2fx
convert
translate
grep
locate
fish
amplicon
duplicate
rmdup
common
split
split2
pair
sample

```
[mej85@login0b PGTB]$ head resources/gfp_barcodes_25bp.fa
>Barcode1
CTGTAGTCTACACGACATCATCTAG
>Barcode2
ATCGTCTACAGTACGAGCGCTCTAC
>Barcode3
ACAGACTGTCTACGACGACTATAGT
>Barcode4
GTACTCGCATGTGATCGCAGTGCAG
>Barcode5
CTGCGCATATCGTACATGTGTCTGA
```

https://bioinf.shenwei.me/seqkit/usage/#locate

HPC - Pitt HTC CRC

- Connecting remotely
  - VPN
  - SSH
- Interacting with resources
  - Bash
    - Language used to move around file system and interact with files
  - Emacs
    - Opening script/file to edit and save
  - Slurm workload manager
    - Submitting scripts/computer programs to a 'node' to run
  - **Conda environments**
    - Package manager for bundling software or tools together to function cohesively

Pre-run checks

- Install Anaconda

```
[mej85@login0b PGTB]$ bash /bgfs/lbyrne/PGTB/resources/Anaconda3-2020.11-Linux-x86_64.sh

Welcome to Anaconda3 2020.11

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

```
Anaconda3 will now be installed into this location:
/ihome/lbyrne/mej85/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/ihome/lbyrne/mej85/anaconda3] >>>
```

Pre-run checks

- Once Anaconda is installed, create conda scaavengr environment

```
[mej85@login0b PGTB]$
[mej85@login0b PGTB]$ conda env create -f /bgfs/lbyrne/PGTB/resources/scaavengr_env.yml
```

- Active env and check that it's working properly
  - Activating loads all software/tools into your working environment

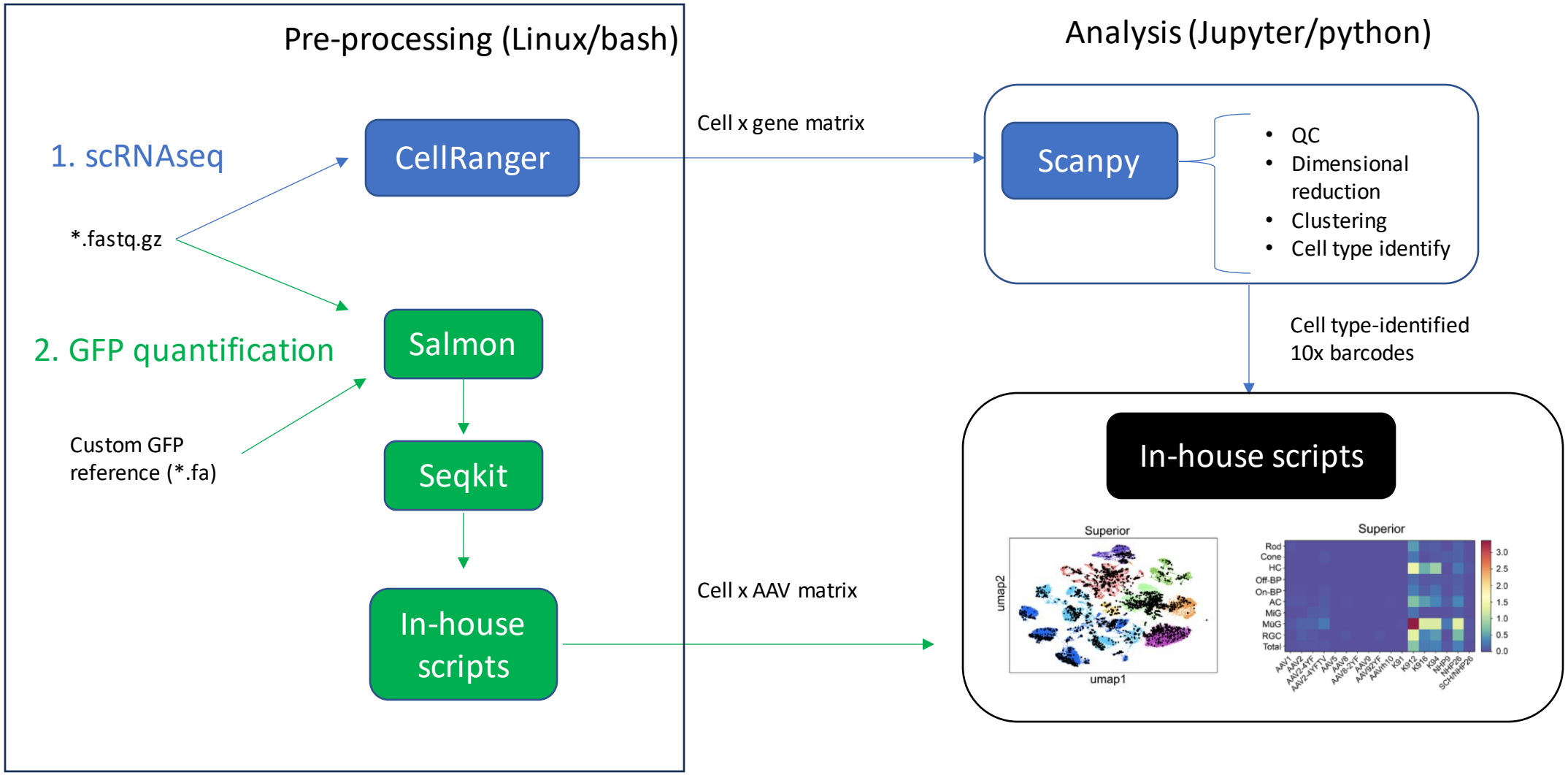
```
[mej85@login0b PGTB]$ conda activate scaavengr_env
(scaavengr_env) [mej85@login0b PGTB]$ python
Python 3.7.12 | packaged by conda-forge | (default, Oct 26 2021, 06:08:53)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from Bio import SeqIO
>>>
>>>
```

While the pipeline is running…
Practice making a script

- Salmon index
  - Set interpreter
  - Simple arguments
    - Set variables through arguments
  - Command
  - Save and execute locally

Lunch Break

scAAVengr pipeline



Running Pre-Processing Pipeline

1. Cellranger
   - Create [gene x cell] matrix
2. Salmon
   - Quantify GFP reads
3. Seqkit
   - Extract GFP barcodes
4. **In-house script**
   - **Create [gfp barcode x cell] matrix**

Cell x GFP matrix

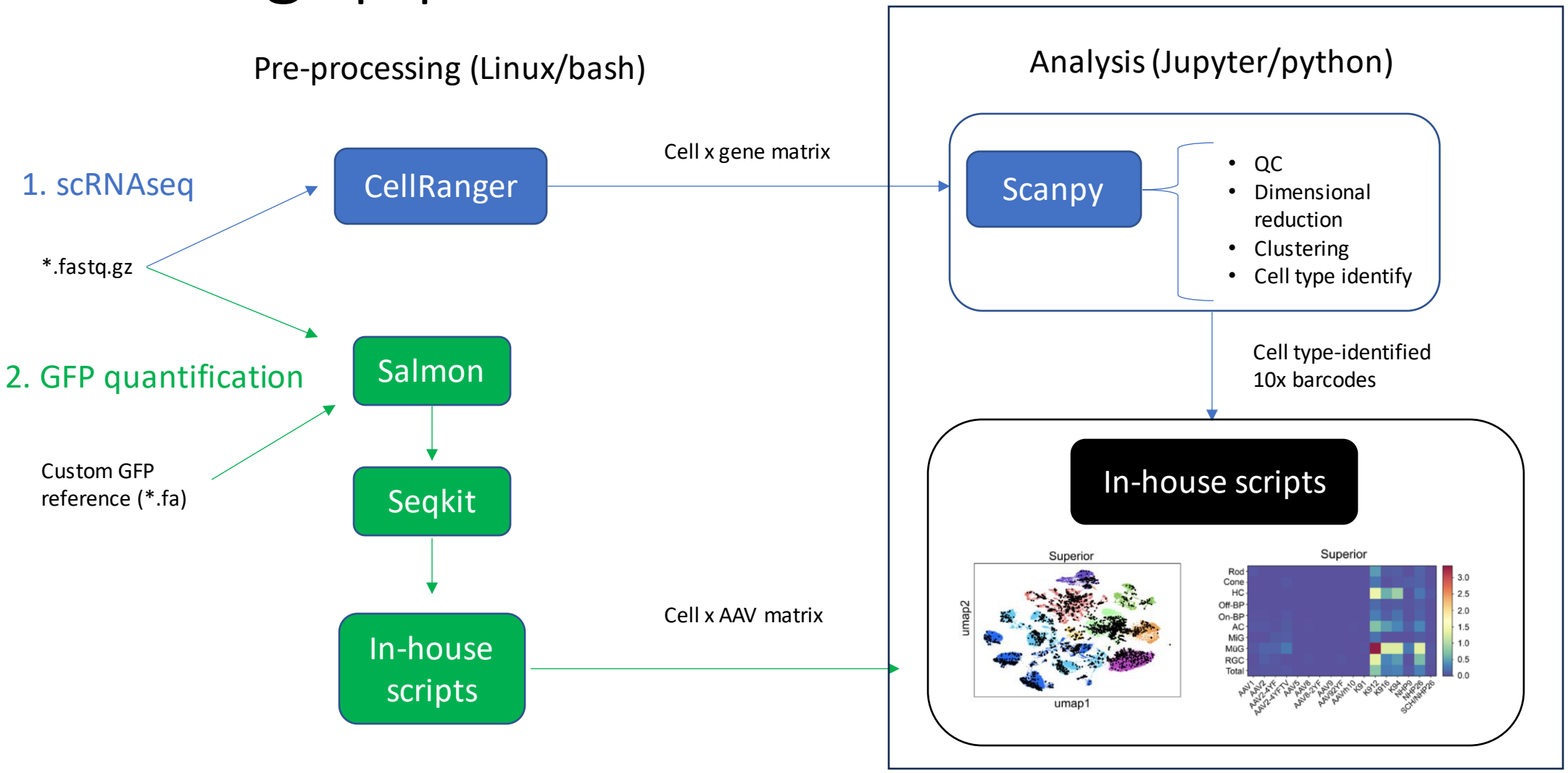
Executing script

```
conda activate scaavengr_env
python ~/git_repos/PGTB/gfp_cell_matrix.py \
    --bc_uniq `pwd`/salmon_quant/LB1_BYR819A1_S1_10x_bc_umi.txt \
    --gfpbc `pwd`/salmon_quant/LB1_BYR819A1_S1_R2_001.subgfp.seqkitlocate.txt \
    --out `pwd`/salmon_quant/LB1_BYR819A1_S1_gfp_cell_matrix.csv \
    --outbin `pwd`/salmon_quant/LB1_BYR819A1_S1_gfp_cell_matrix.binary.csv
```

Code within the script itself

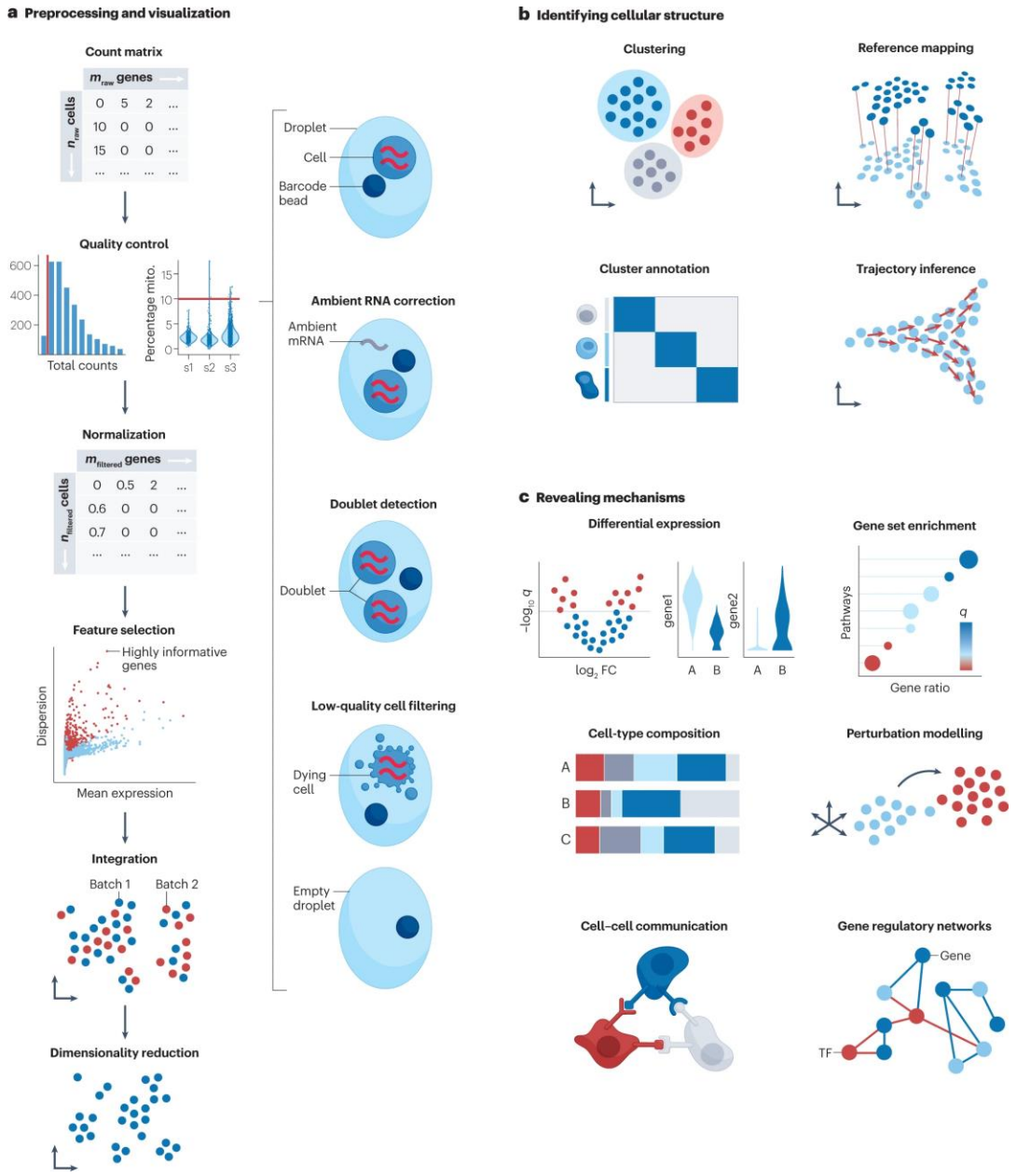
```
#!/usr/bin/env python

import argparse
import array
import numpy as np
import pandas as pd
import sys

################ MAIN

# GET OPTIONS
parser = argparse.ArgumentParser()
parser.add_argument('--bc_uniq')
parser.add_argument('--gfpbc')
parser.add_argument('--out')
parser.add_argument('--outbin')
args = parser.parse_args()
```

scAAVengr pipeline



ScRNAseq Analysis

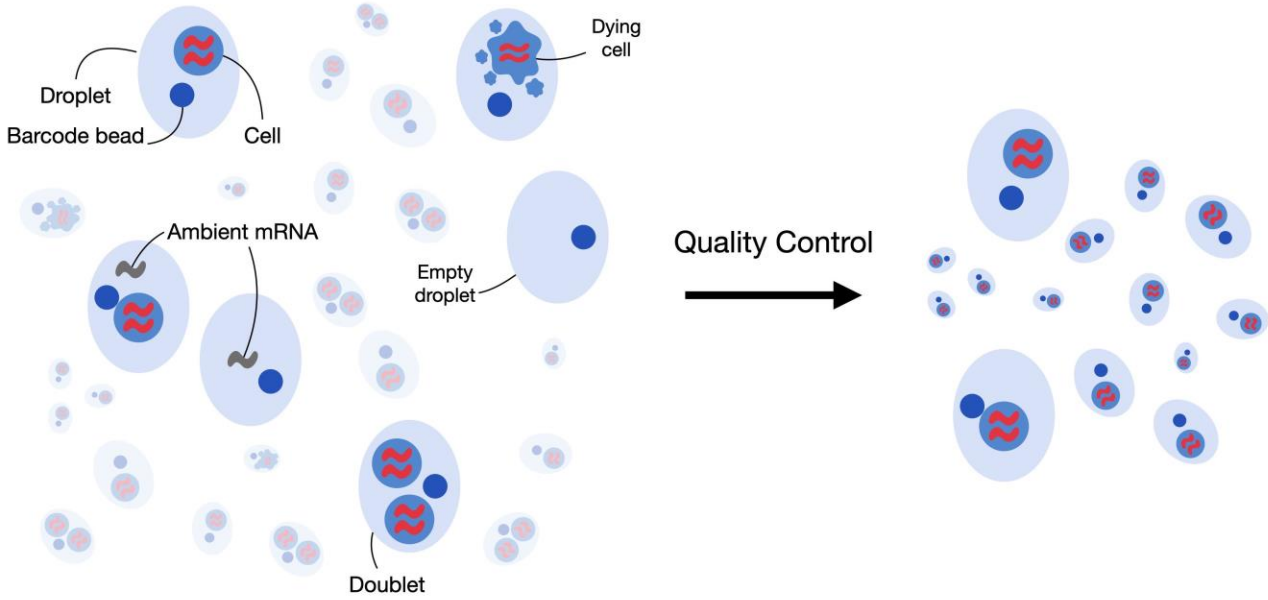
- Quick analysis of scRNAseq data to identify cell types
  - Many resources/guides that cover more in-depth single cell techniques:
    - www.sc-best-practices.org
    - Heumos, L. *et al.* Best Practices for single-cell analysis across modalities. *Nature* (2023).
- Today's focus will be on quickly getting cell types so we can get to the scAAVengr heatmap visualization



Heumos, L. *et al.* Best Practices for single-cell analysis across modalities. *Nature* (2023).

QC

- Empty Droplets
  - Detecting 10x barcodes/droplets that did not contain a full cell, but rather a mixture of free-floating RNA
  - Can occur from cell lysis during processing
  - Tools: EmptyDrops
- Doublets
  - Detecting 10x barcodes/droplets that contained more than 1 cell
  - Can occur from overloading 10x channel
  - Tools: Vaeda (see Hannah!), scds, etc.
- Ambient RNA
  - Background RNA contamination captured randomly in 10x barcodes/droplets
  - Can occur from cell lysis during processing
  - Tools: SoupX
- Normalization (pre-processing)
  - Corrects for variance in count/read depth
  - Tools: Scran

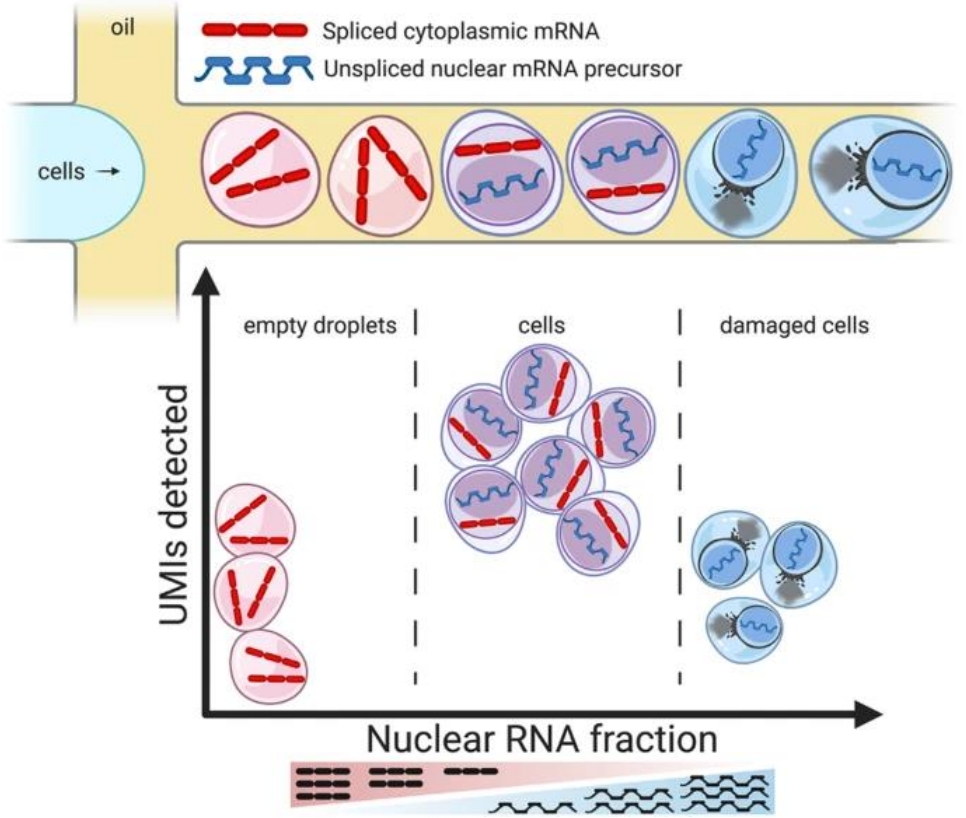


https://www.sc-best-practices.org/preprocessing_visualization/quality_control.html

QC

- Empty Droplets
  - Detecting 10x barcodes/droplets that did not contain a full cell, but rather a mixture of free-floating RNA
  - Can occur from cell lysis during processing
  - Tools: EmptyDrops, DropletQC

- Method (EmptyDrops)
  - Estimate ambient RNA profile
  - Significant deviations from this profile are considered genuine cells

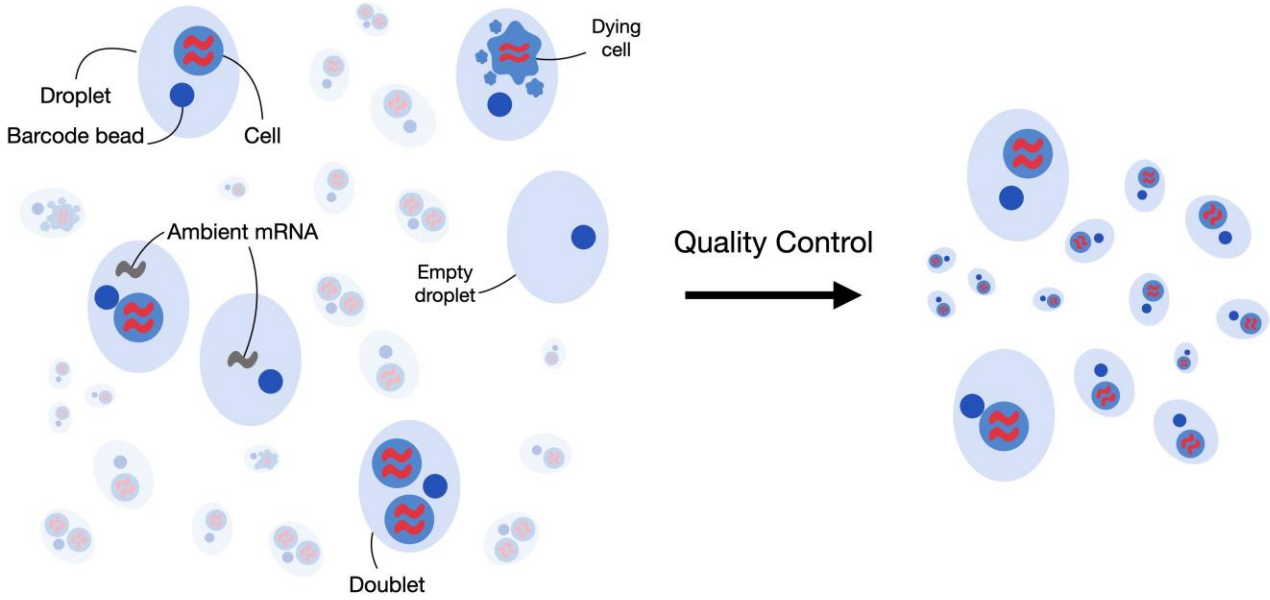


Muskovic, W., Powell, J.E. DropletQC: improved identification of empty droplets and damaged cells in single-cell RNA-seq data. *Genome Biol* (2021).

QC

- Doublet detection
  - Detecting 10x barcodes/droplets that contained more than 1 cell
  - Can occur from overloading 10x channel
  - Tools: Vaeda (see Hannah!), scds, scrublet, etc.

- Method (scds)
  - Co-expression based doublet scoring
  - Assumes doublets containing two different cell type will co-express two different sets of genes that are usually not active in the same cell

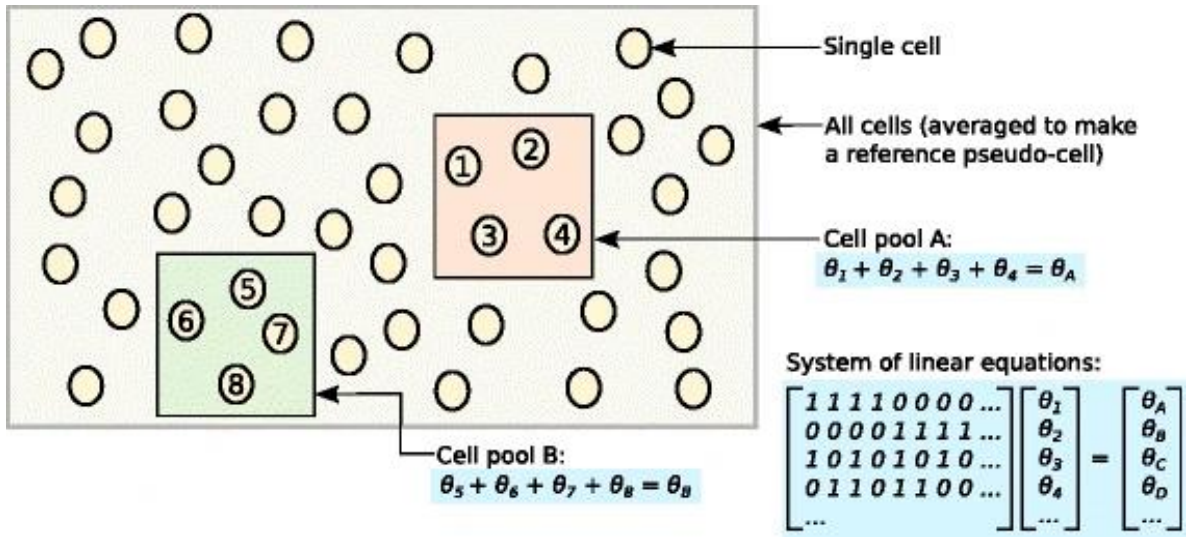


https://www.sc-best-practices.org/preprocessing_visualization/quality_control.html

Pre-processing

- Normalization (pre-processing)
  - Corrects for variance in count/read depth
  - Tools: Scran

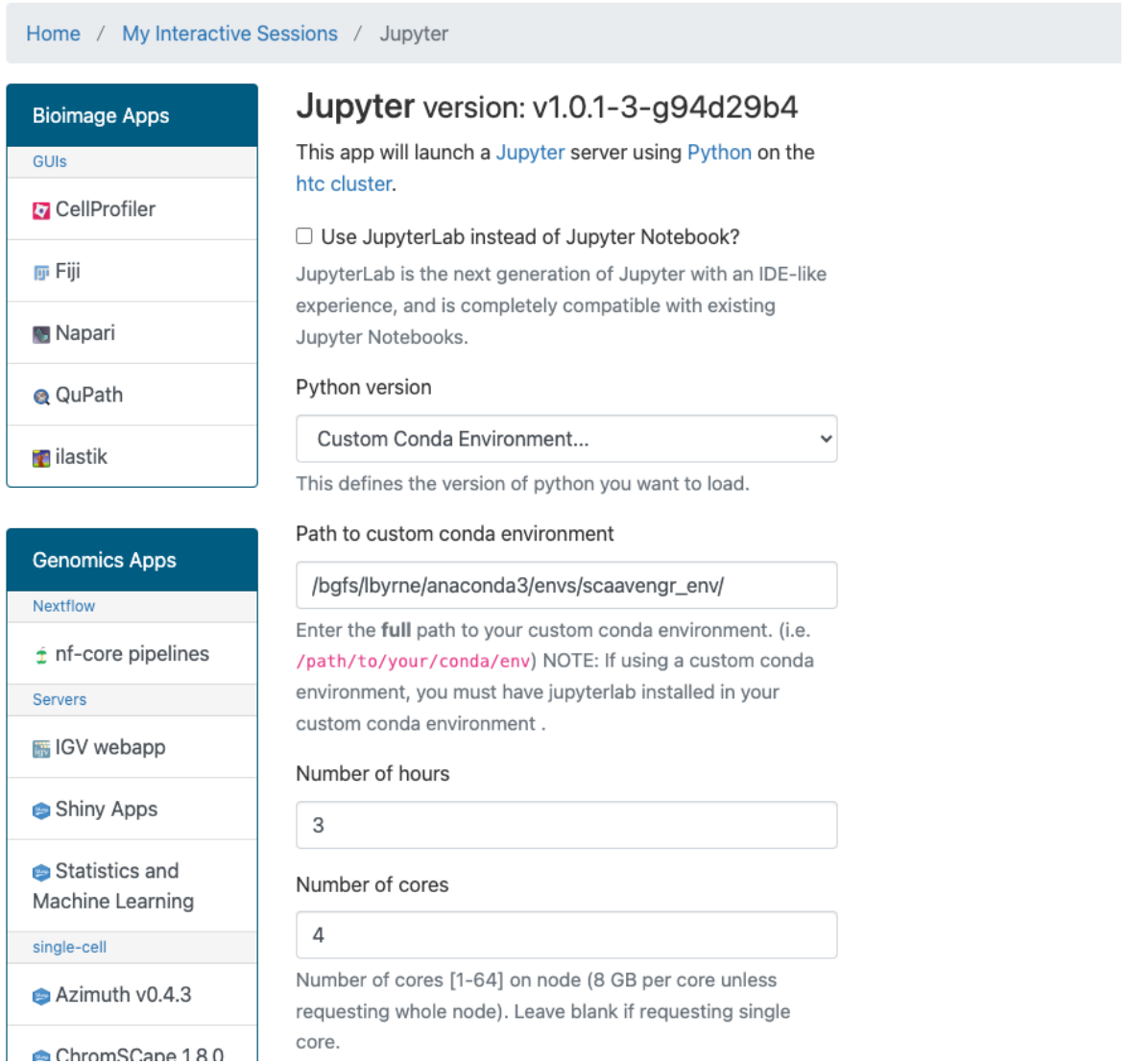
- Method
  - Pool cells together with similar count depth
  - Estimate pool-based size factor using linear regression over genes
  - Performs well for batch correction effects



Lun L., et al. Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biol* (2016).

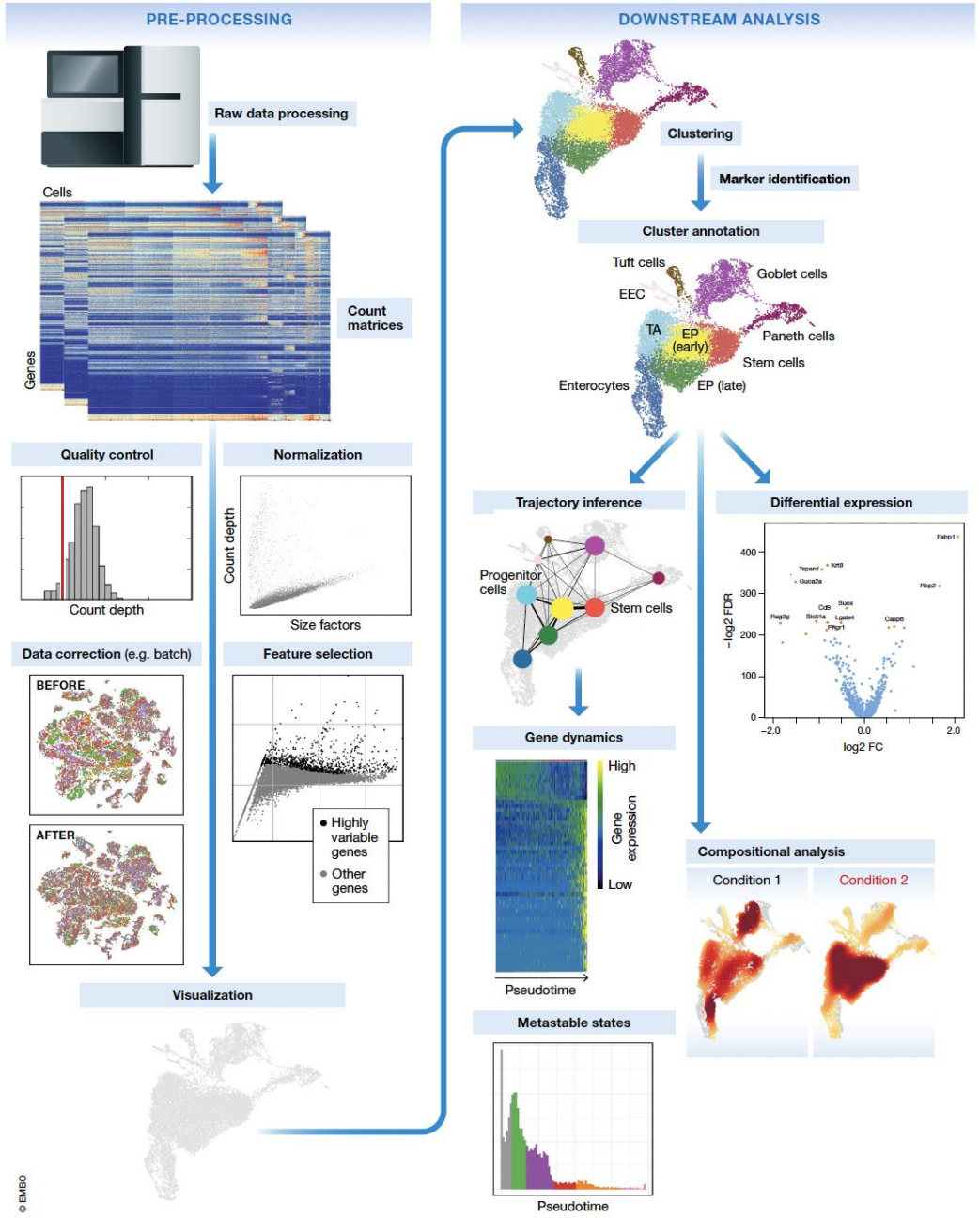
Jupyter Notebook

- Navigate to
  - https://ondemand.htc.crc.pitt.edu/
  - Interactive Apps > Jupyter
- Use custom conda environment
  - Path to scaavengr env
- 4 hours, 4 cores
- Click 'Launch'



Analysis

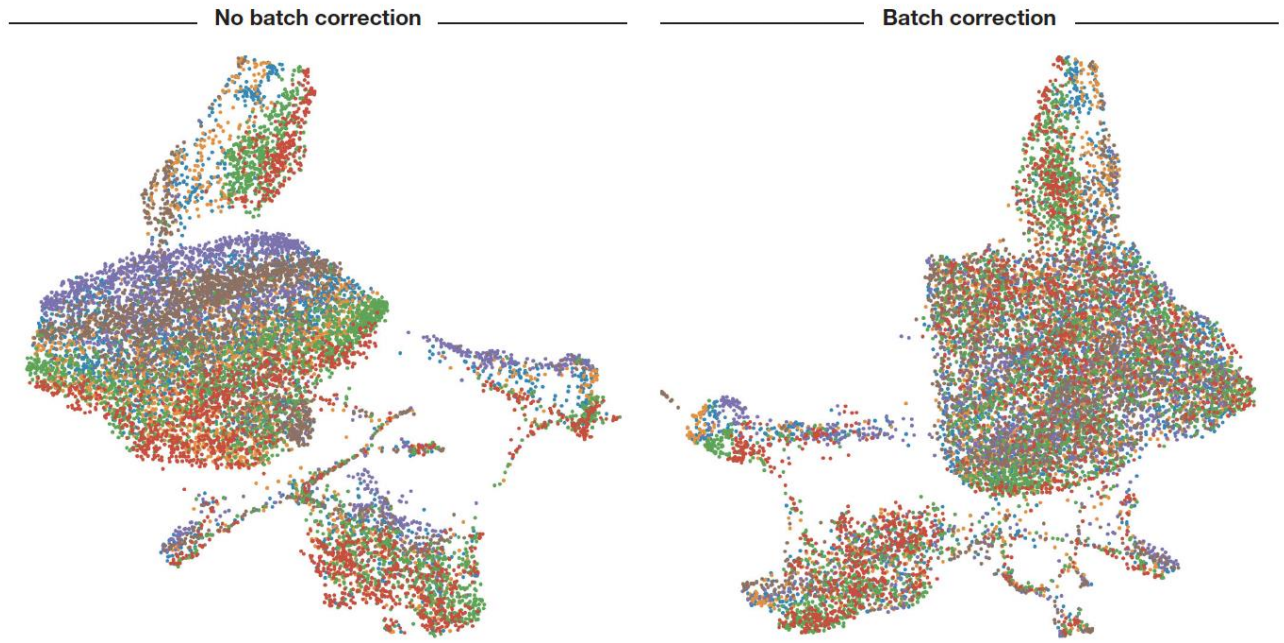
- Batch correction
  - Regressing out differences across samples
- Dimensional reduction
  - Downsizing dataset to capture most variable/signifcant components
- Clustering
  - Grouping together similar cells
- Cluster annotation
  - Identify cell types with differential expression of marker genes



Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019

Analysis

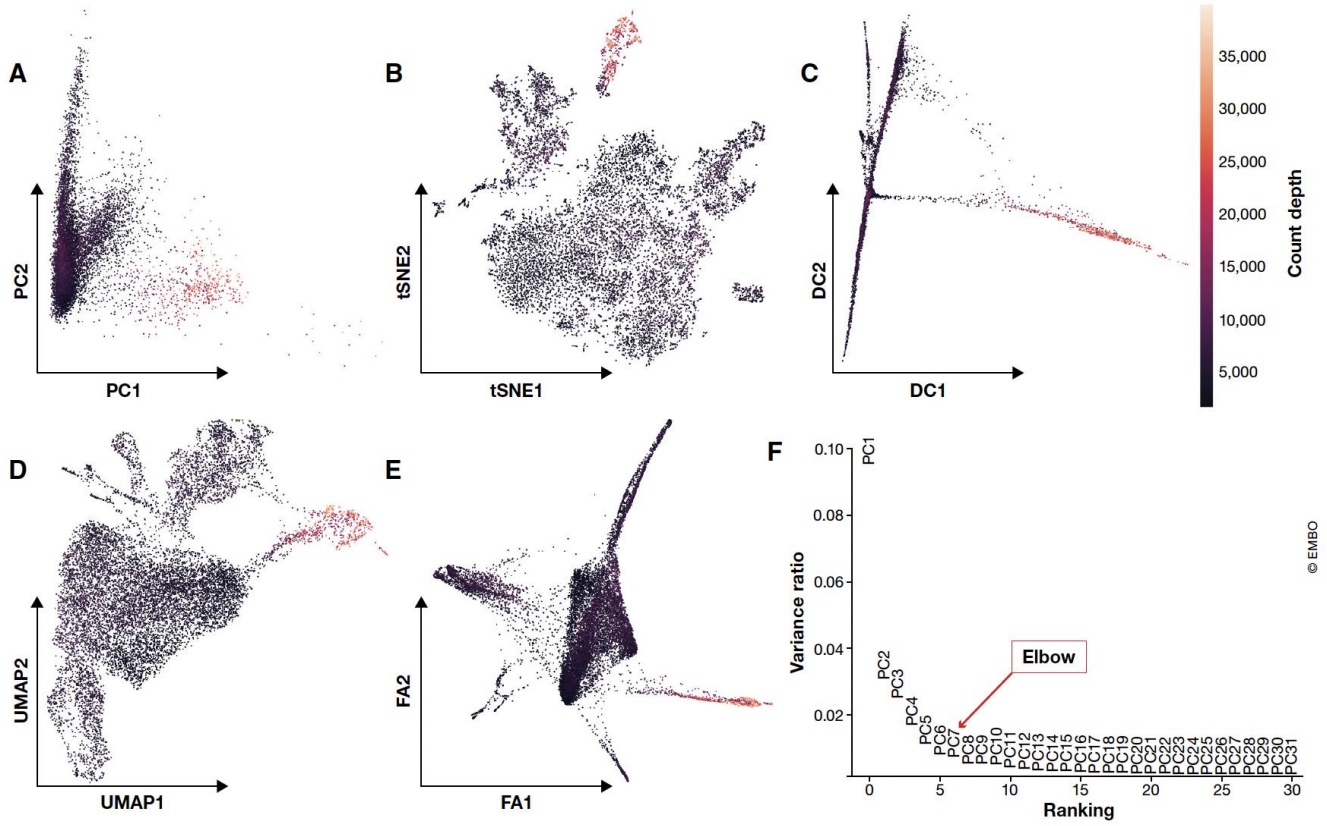
- Batch correction
  - Regress out biological/technical artifacts across samples
  - Tool: Harmony
- Method
  - Cluster cells in low dimensionnal space and favor shared/similar profiles across datasets; correct based on these computed factors



Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019

Analysis

- Dimensional reduction
  - Feature selection/visualization
  - Used for visualization (2 or 3 dimensions) or summarizing the data (multiple significant dimensions)
  - Tools: PCA, UMAP
- Method
  - Reduced dimensions are created through linear or non-linear combinations of gene expression vectors

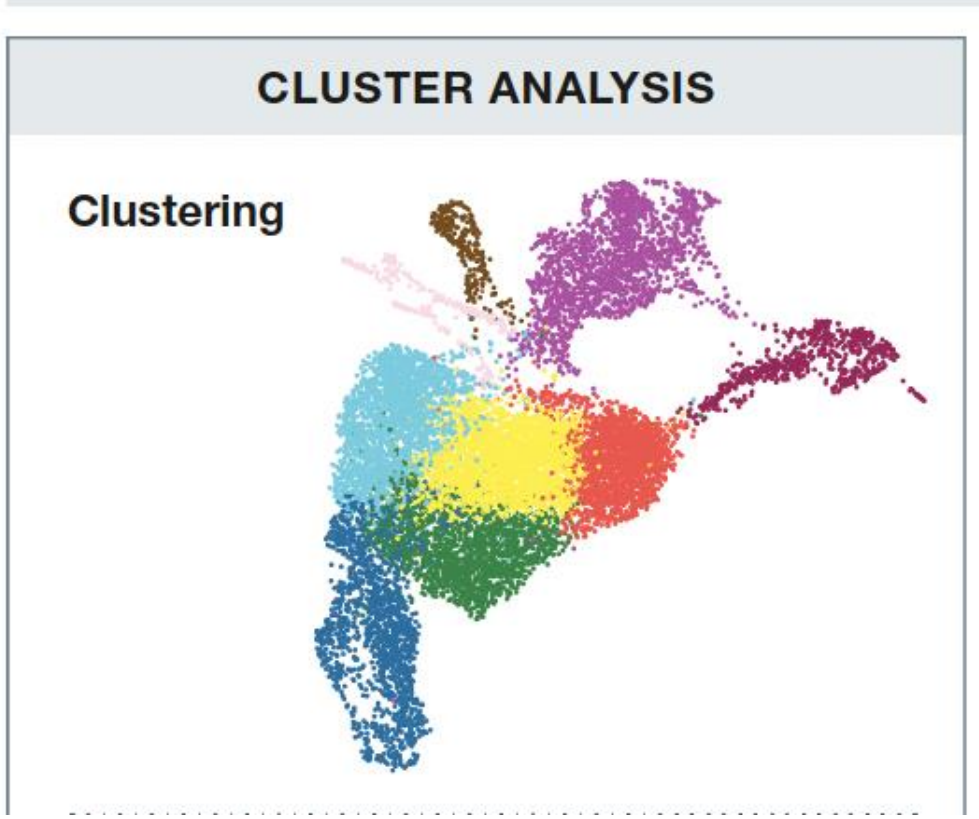


Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019

Analysis

- Clustering
  - Group cells together based on similarity of gene expression profiles
  - Tools: Leiden, Louvain

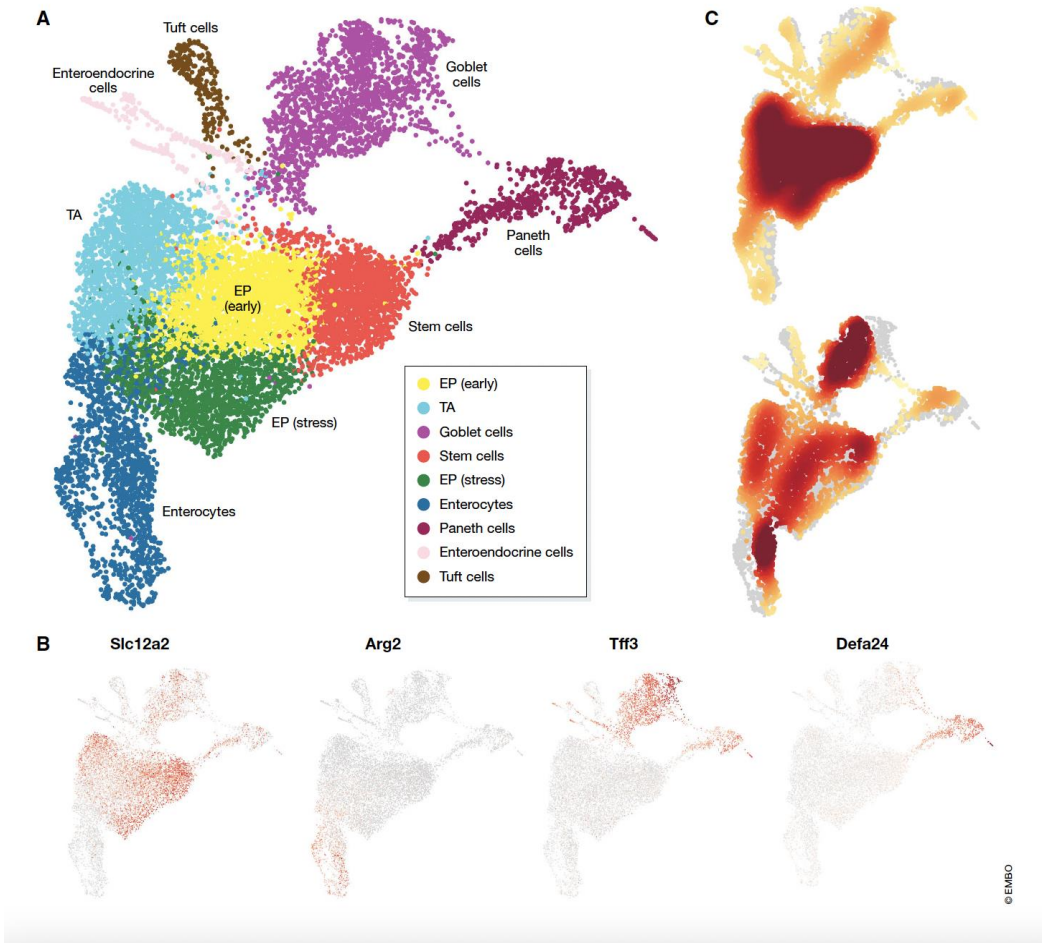
- Method
  - Graph representation of cells, connected to other cells (K nearest neighbors 'KNN') using euclidean distance in low dimensional space



Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019

Analysis

- Cell type annotation
  - Identify cell types based on differential gene expression of marker genes
  - Tools: t-test
- Method
  - Relies on external datasets describing expected expression profiles of cell types



Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019

AAV heatmaps/plots