

# Genome 540 Discussion

Conor Camplisson

February 14<sup>th</sup>, 2023

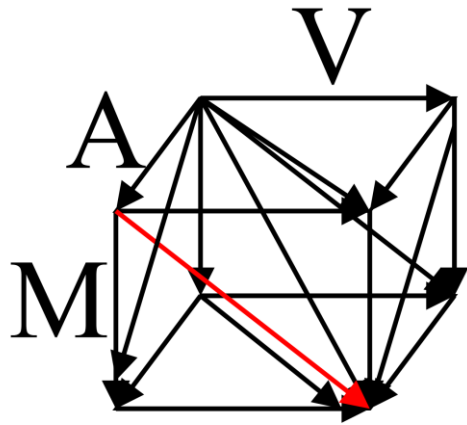
# Outline

- Homework 5 wrap-up
- Related topics:
  - JupyterLab IDE on GS cluster
  - GitHub basics
  - Virtual environments and conda
  - Intro to Snakemake
- Homework 6 questions

# Outline

- Homework 5 wrap-up
- Related topics:
  - JupyterLab IDE on GS cluster
  - GitHub basics
  - Virtual environments and conda
  - Intro to Snakemake
- Homework 6 questions

# Homework 5 Wrap-up



Your program should output the following:

1. The maximum path score
2. A list of all edge weights (sorted alphabetically by edge name)
3. A histogram of edge counts (again, sorted alphabetically by edge name)
4. The highest-scoring alignment, formatted vertically (as described above)

Assignment: GS540 HW5  
 Name: {YOURNAME}  
 Email: {YOUREMAIL}  
 Language: {YOURLANGUAGE}  
 Runtime: {YOURRUNTIME}

Score: 82.0

Edge weights:

```
--A = -12
--C = -12
--D = -12
--E = -12
--F = -12
```

```
.
.
.
list all edge weights in alphabetical order
(only first/last 5 shown here)
```

```
.
.
.
 YYS = 3
  YYT = 3
  YYV = 5
  YYW = 11
  YYY = 21
```

Edge counts:

```
--A = 8832
--C = 17664
--D = 52992
--E = 70656
--F = 44160
```

```
.
.
.
list all the edge counts in alphabetical order
(only first/last 5 shown here)
```

```
.
.
.
 YYS = 48
  YYT = 24
  YYV = 72
  YYW = 24
  YYY = 60
```

Local alignment:

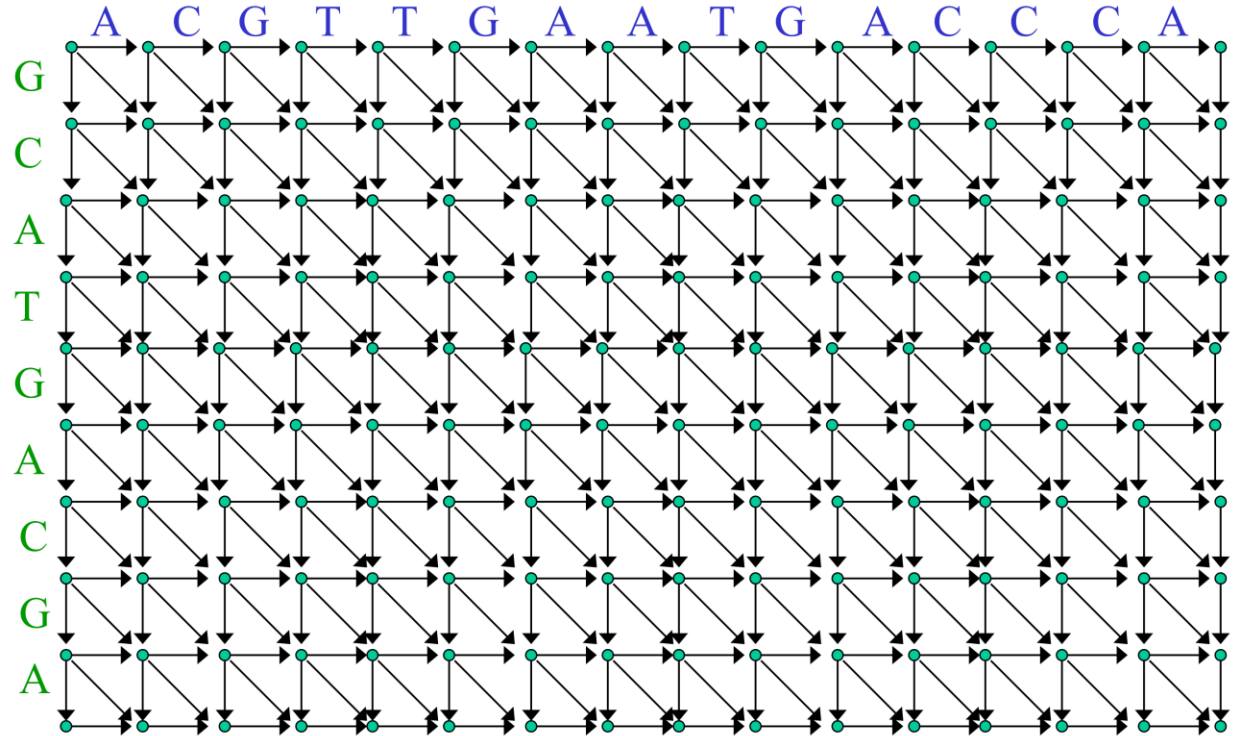
```
KKK
DLK
YWY
G--
LFL
KVN
REH
IPI
QRQ
KPN
SNS
AVV
FFF
MVE
GSG
SGE
LVI
KKS
KDE
HSP
AVQ
LAI
NQL
STN
LVL
LVK
TDY
DYE
LLL
RLN
RQK
MYV
VTI
--R
--K
NPD
VVI
DED
SSS
-G-
VLV
IMI
VLL
FFF
```

# Homework 5 Wrap-up

1 seq  $\rightarrow$  1-D sequence graph



2 seqs  $\rightarrow$  2-D Edit graph(pairwise alignment)



# Outline

- Homework 5 wrap-up
- Related topics:
  - JupyterLab IDE on GS cluster
  - GitHub basics
  - Virtual environments and conda
  - Intro to Snakemake
- Homework 6 questions

# Outline

- Homework 5 wrap-up
- **Related topics:**
  - JupyterLab IDE on GS cluster
  - GitHub basics
  - Virtual environments and conda
  - Intro to Snakemake
- Homework 6 questions

# Jupyterlab IDE on GS Cluster

The screenshot displays the JupyterLab interface. On the left is a file browser showing the directory structure: `/ workspace / PaintSHOP_resources /` with files `LICENSE`, `PaintSHOP-logo.png`, and `README.md`. The central editor shows the `README.md` file with HTML code for a table of FISH Probe Sets. The right pane shows the rendered HTML output, featuring the PaintSHOP logo and a table of resources. The terminal at the bottom shows the output of the `q` command.

```
1 <div align="center">
2   <a href="#readme"></a>
3 </div>
4
5 # PaintSHOP_resources
6
7 A collection of downloadable resources accompanying the PaintSHOP application.
8
9 ## FISH Probe Sets
10
11 <div align="center">
12   <table>
13     <thead>
14       <tr>
15         <th align="center">Assembly</th>
16         <th align="center">DNA FISH probes</th>
17         <th align="center">RNA FISH probes<br>(isoform-resolved)</th>
18         <th align="center">RNA FISH probes<br>(isoform-flattened)</th>
19       </tr>
20     </thead>
21     <tbody>
22       <tr>
23         <td align="left">hg38 newBalance</td>
24         <td align="center"><a href="https://paintshop-
25 bucket.s3.amazonaws.com/v1.2/resources/all/hg38_all_newBalance.zip">download</a>
26         <td align="center"><a href="https://paintshop-
27 bucket.s3.amazonaws.com/v1.2/resources/refseq/hg38_refseq_newBalance.zip">downlo
28 ad</a></td>
29         <td align="center"><a href="https://paintshop-
30 bucket.s3.amazonaws.com/v1.2/resources/iso/hg38_iso_newBalance.zip">download</a>
31 </td>
32       </tr>
33     </tbody>
34   </table>
35 </div>
```

**PaintSHOP\_resources**

A collection of downloadable resources accompanying the PaintSHOP application.

**FISH Probe Sets**

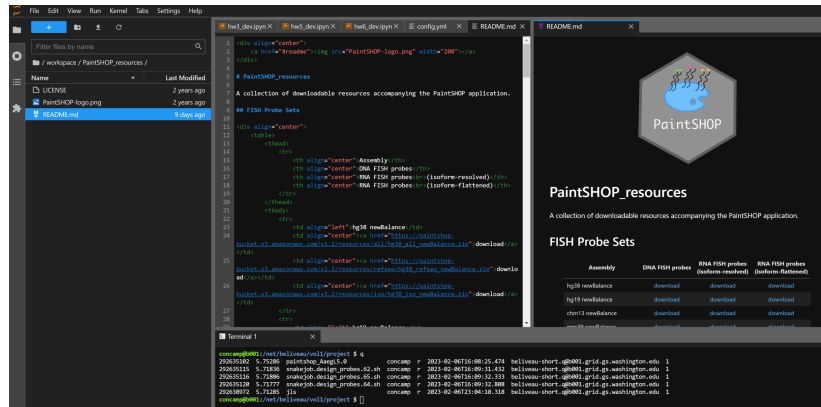
Assembly	DNA FISH probes	RNA FISH probes (isoform-resolved)	RNA FISH probes (isoform-flattened)
hg38 newBalance	download	download	download
hg19 newBalance	download	download	download
chm13 newBalance	download	download	download
mm38 newBalance	download	download	download

```
concamp@b001:/net/beliveau/vol1/project $ q
292635102 5.75286 paintshop_AaegL5.0 concamp r 2023-02-06T16:08:25.474 beliveau-short.q@b001.grid.gs.washington.edu 1
292635115 5.71836 snakejob.design_probes.62.sh concamp r 2023-02-06T16:09:31.432 beliveau-short.q@b001.grid.gs.washington.edu 1
292635116 5.71806 snakejob.design_probes.65.sh concamp r 2023-02-06T16:09:32.333 beliveau-short.q@b001.grid.gs.washington.edu 1
292635120 5.71777 snakejob.design_probes.64.sh concamp r 2023-02-06T16:09:32.808 beliveau-short.q@b001.grid.gs.washington.edu 1
292638972 5.71285 jls concamp r 2023-02-06T23:04:10.318 beliveau-short.q@b001.grid.gs.washington.edu 1
concamp@b001:/net/beliveau/vol1/project $
```

Web browser interface to GS cluster



# Jupyterlab IDE on GS Cluster



JupyterLab is a web app:

- A webserver runs the application
- A user goes to the app's URL
- The server renders the homepage (the IDE)

1. Log into the cluster and run a JupyterLab server

```
(on grid) $ jupyter lab --no-browser --ip=$(hostname) --port=8889
```

- App is running on \$HOST:\$PORT

2. Create an SSH tunnel from your PC to \$HOST on \$PORT

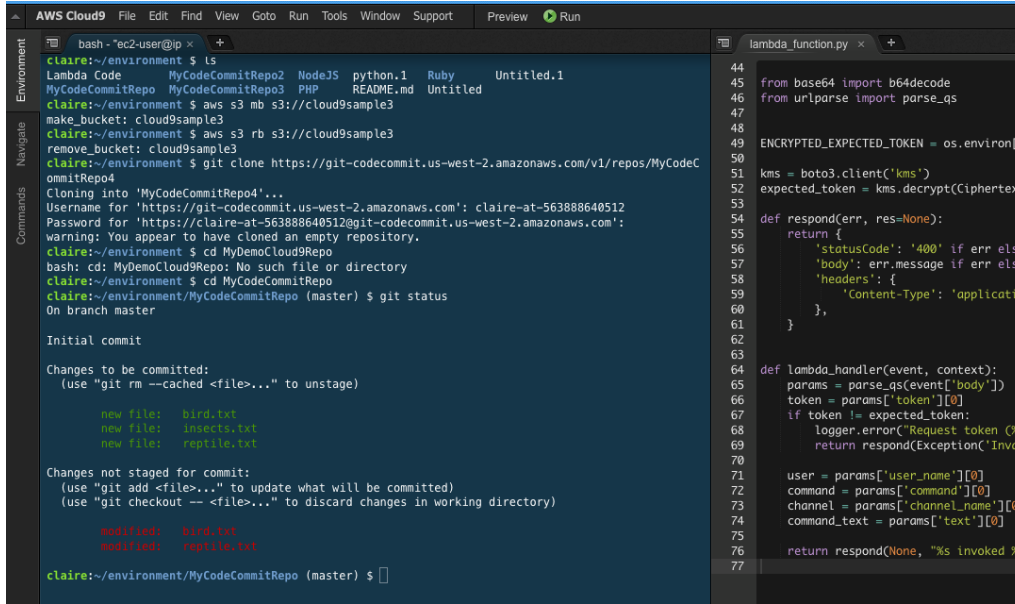
```
(local) $ ssh -L 8889:sage001:8889 concamp@nexus.gs.washington.edu
```

3. Access the IDE at localhost:\$PORT in your web browser

```
(local) http://localhost:8889/
```

Contact me for a detailed guide!

# AWS Cloud9: Similar Solution (for non-UWGS servers)



```
bash - ec2-user@ip x +
Environment
  claire:~/environment $ ls
  Lambda Code      MyCodeCommitRepo2  NodeJS  python.1  Ruby  Untitled.1
  MyCodeCommitRepo MyCodeCommitRepo3  PHP     README.md  Untitled
  claire:~/environment $ aws s3 mb s3://cloud9sample3
  make_bucket: cloud9sample3
  claire:~/environment $ aws s3 rb s3://cloud9sample3
  remove_bucket: cloud9sample3
  claire:~/environment $ git clone https://git-codecommit.us-west-2.amazonaws.com/v1/repos/MyCodeCommitRepo4
  Cloning into 'MyCodeCommitRepo4'...
  Username for 'https://git-codecommit.us-west-2.amazonaws.com': claire-at-563888640512
  Password for 'https://claire-at-563888640512@git-codecommit.us-west-2.amazonaws.com':
  warning: You appear to have cloned an empty repository.
  claire:~/environment $ cd MyDemoCloud9Repo
  bash: cd: MyDemoCloud9Repo: No such file or directory
  claire:~/environment $ cd MyCodeCommitRepo
  claire:~/environment/MyCodeCommitRepo (master) $ git status
  On branch master

  Initial commit

  Changes to be committed:
    (use "git rm --cached <file>..." to unstage)

    new file:   bird.txt
    new file:   insects.txt
    new file:   reptile.txt

  Changes not staged for commit:
    (use "git add <file>..." to update what will be committed)
    (use "git checkout -- <file>..." to discard changes in working directory)

    modified:  bird.txt
    modified:  reptile.txt

  claire:~/environment/MyCodeCommitRepo (master) $

lambda_function.py x +
44
45 from base64 import b64decode
46 from urlparse import parse_qs
47
48
49 ENCRYPTED_EXPECTED_TOKEN = os.environ['
50
51 kms = boto3.client('kms')
52 expected_token = kms.decrypt(Ciphertext
53
54 def respond(err, res=None):
55     return {
56         'statusCode': '400' if err else
57         'body': err.message if err else
58         'headers': {
59             'Content-Type': 'applicati
60     },
61 }
62
63
64 def lambda_handler(event, context):
65     params = parse_qs(event['body'])
66     token = params['token'][0]
67     if token != expected_token:
68         logger.error("Request token (%s
69         return respond(Exception("Inval
70
71     user = params['user_name'][0]
72     command = params['command'][0]
73     channel = params['channel_name'][0]
74     command_text = params['text'][0]
75
76     return respond(None, "%s invoked %s
77
```



- Provides web browser IDE interface to any server you can SSH into
- Fast access to any AWS data, cloud compute infrastructure
  - Analogous to doing dev right on GS cluster
- Cloud9 IDE itself is free to use, even on a non-AWS server
- Supports collaboration, live co-editing of code, backed by AWS Auth layer

# Outline

- Homework 5 wrap-up
- **Related topics:**
  - JupyterLab IDE on GS cluster
  - **GitHub basics**
  - Virtual environments and conda
  - Intro to Snakemake
- Homework 6 questions

# GitHub Basics

## Front-end rendering of Markdown documentation

```
# PaintSHOP Pipeline
```

```
[[Snakemake](./docs/img/snakemake.svg)](https://snakemake.r  
[[DOI](./docs/img/preprint.svg)](https://doi.org/10.1101/2020.07.05.188797)
```

```
## Overview
```

```
[PaintSHOP](https://www.biorxiv.org/content/10.1101/2020.07.05.188797) is a technology that enables the interactive design of oligonucleotide transcriptome-scale and is comprised of two components:
```

1. A scalable machine learning pipeline for probe specificity prediction
2. An interactive Shiny web application for probe design

```
This repository contains the Snakemake workflow for the machine learning needed to design probes for additional genomes not already hosted on the web application.
```

```
## Installation
```

1. Make sure you have [conda](https://docs.conda.io/en/latest/) installed.

### PaintSHOP Pipeline

snakemake >5.20.1 DOI: 10.1101/2020.07.05.188797

#### Overview

PaintSHOP is a technology that enables the interactive design of oligonucleotide transcriptome-scale and is comprised of two components:

1. A scalable machine learning pipeline for probe specificity prediction
2. An interactive Shiny web application for probe design

This repository contains the Snakemake workflow for the machine learning needed to design probes for additional genomes not already hosted on the web application.

#### Installation

1. Make sure you have [conda](#) installed.

## Jupyter notebook rendering

main repeat\_suppression / notebooks / 20221031\_preprocess\_df\_canonica

conorcampbell add preprocess notebook

1 contributor

298 lines (298 sloc) | 10.2 KB

```
In [1]: import pandas as pd

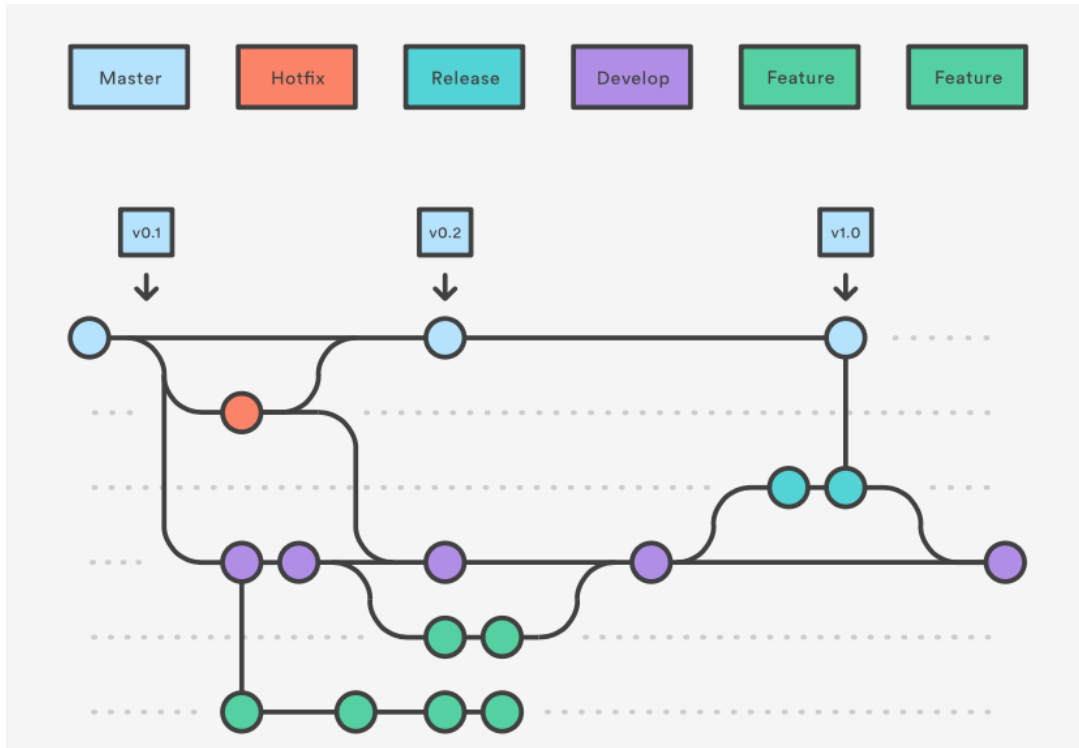
def rev_comp(seq):
    '''Return the reverse complement of the input DNA sequence.'''
    rc_dict = {
        'A': 'T',
        'C': 'G',
        'G': 'C',
        'T': 'A',
    }
    rc_seq = ''.join(rc_dict[base] for base in seq[::-1])
    return rc_seq
```

## Front-end features:

- Browse, search codebase, link to a line of code
- Open an issue (to alert developers, comment, track updates)
- Examine 'diffs' between each version
- Homepage and documentation for your project
- Front-end rendering (notebooks, pdf, markdown), syntax highlighting (code)

# GitHub Basics

## “Git flow” branching model



Very useful when a project:

- Is in production, with active users
- Has multiple developers
- Has a large codebase
- Has a complicated dev life cycle:
  - Production version live
  - Next version being developed
  - Push a bug fix to dev and prod branches

...but can be cumbersome/confusing

You can go very far (building and deploying projects) with just:

```
$ git status
```

```
$ git add <file(s)>
```

```
$ git commit -m 'fix a rounding bug'
```

```
$ git push
```

# GitHub Basics

## Only once ever (per server)

Set up your SSH keys with GitHub

- Generate SSH keys on your machine if you haven't already
- Using the GitHub website, add your public key to your account
- Now that machine can log into GitHub as you to push/pull code

## When starting a new project

- Create a new repo in the GitHub web interface
  - The auto README.md, LICENSE, .gitignore features are very useful
- Clone the new repo to your machine using the command line
- Start working in the new repo locally

## Day to day dev workflow with GitHub

- Develop and test some code locally
- When you've implemented a coherent "change" (one or several files), "commit" your change(s) with a descriptive message
- Push your latest commit(s) to the cloud repo to update it

# Outline

- Homework 5 wrap-up
- **Related topics:**
  - JupyterLab IDE on GS cluster
  - GitHub basics
  - **Virtual environments and conda**
  - Intro to Snakemake
- Homework 6 questions

# Virtual Environments & Conda

Why use virtual environments?

Problem: you work on two projects on your PC. One requires numpy version 19 and one requires numpy version 20. *Which version of numpy should you install?*

Solution: use a different virtual env for each project, with the right numpy versions

Problem: your code runs on your machine, but it depends on several locally installed packages. *How will you run it on the cluster? How will your users run it?*

Problem: your app runs in the cloud, and you edit a local version, you are actively adding features and new dependencies often, so both servers keep needing new packages installed. *How can you keep all server(s) up to date?*

Solution: environment.yml in the repo describes a virtual env, when you pull changes (`$ git pull`) you can also `$ conda env update -f environment.yml`



# Virtual Environments & Conda

## Only once ever (per server)

- Install `conda` (e.g. `miniconda3`)
- (optional, faster/powerful) install `mamba` to your `conda` base env
  - From now on, use `$ mamba` instead of `$ conda` executable

## When starting a new project

- Create a new `conda` environment, install dependencies manually
- Export environment to `.yml`
  - only include top-level dependencies (i.e. things you `import`)
  - Use `--no-builds` flag to enable cross-platform env builds from `yaml`

## Day to day dev workflow with GitHub

- When working in a repo, activate that repo's `conda` env
- If you install and use a new package, add it to `.yml` and push to cloud repo
  - Before you ever push an edited `.yml`, make sure the env actually builds

# Outline

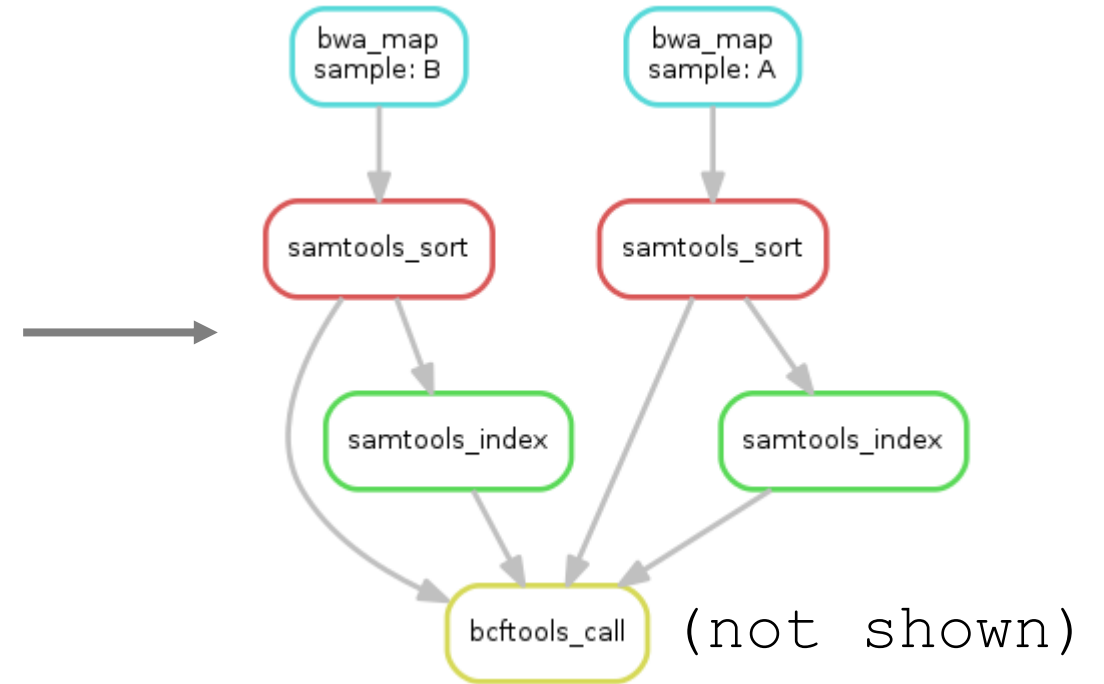
- Homework 5 wrap-up
- **Related topics:**
  - JupyterLab IDE on GS cluster
  - GitHub basics
  - Virtual environments and conda
  - **Intro to Snakemake**
- Homework 6 questions

# Intro to Snakemake

```
rule bwa_map:
  input:
    "data/genome.fa",
    "data/samples/{sample}.fastq"
  output:
    "mapped_reads/{sample}.bam"
  shell:
    "bwa mem {input} | samtools view -Sb - > {output}"

rule samtools_sort:
  input:
    "mapped_reads/{sample}.bam"
  output:
    "sorted_reads/{sample}.bam"
  shell:
    "samtools sort -T sorted_reads/{wildcards.sample} "
    "-O bam {input} > {output}"

rule samtools_index:
  input:
    "sorted_reads/{sample}.bam"
  output:
    "sorted_reads/{sample}.bam.bai"
  shell:
    "samtools index {input}"
```



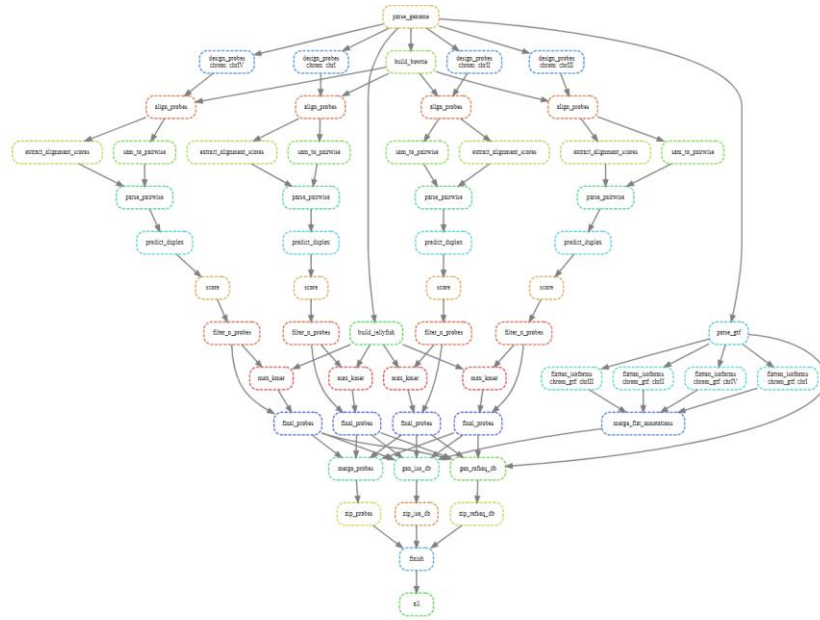
# Intro to Snakemake

Why use virtual Snakemake?

Bash pipelines work. Snakemake just automates several tedious aspects:

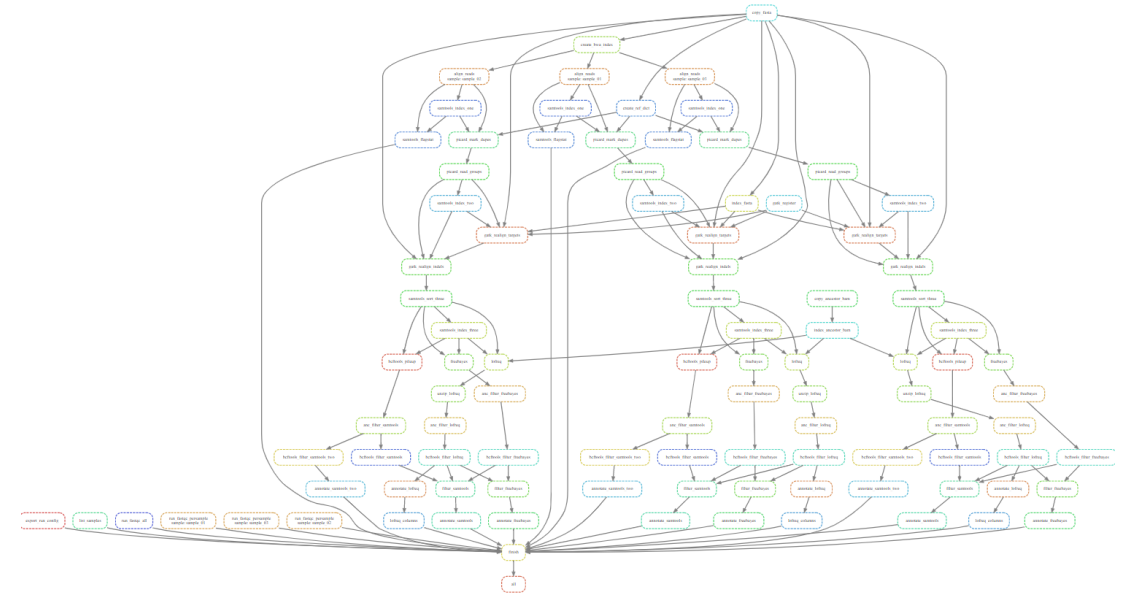
- Automatic creation of target output directories, useful flags for temp files with (optional) auto-delete
- Many utils for logging, benchmarking resource usage, reporting, etc.
- `--cluster` switch for local (1 CPU) vs. parallel (cluster nodes) execution
  - edit pipeline locally, push to cluster/cloud, run same code at scale!
- Move on to next step, per file, as soon as it's available
  - Job dependency graph more efficient than iteration (resource utilization)
  - Listen for target output file creation asynchronously, start next job

# Next time: Simple pipeline in Snakemake



## PaintSHOP Pipeline

Snakemake pipeline for genome-scale mining of optimal homology sequences for [PaintSHOP](#)



## yEvo Pipeline

Variant calling Snakemake pipeline for [yEvo](#) sequencing data



# Example: GitHub + Conda + Snakemake

## Installation

1. Make sure you have [conda](#) installed.
2. Install Mamba to facilitate snakemake installation, as recommended in the [Snakemake docs](#).

```
$ conda install -n base -c conda-forge mamba
```

3. Clone this repo, then create and activate the provided [environment](#):

```
$ git clone https://github.com/beliveau-lab/PaintSHOP_pipeline.git \  
&& cd PaintSHOP_pipeline/ \  
&& mamba env create -f environment.yml \  
&& conda activate paintshop_snakemake
```

## Running the pipeline

A complete example is included to test the pipeline installation. To run the pipeline on the included sample files:

```
$ cd example_run/ && ./run_pipeline.sh
```

When this example is run, pipeline output will be generated [here](#). Expected outputs are provided [here](#) for comparison.

To run the pipeline on your own data, update the file paths in [config.yml](#) with the paths to your genome assembly files. For more information on input and output files, see the [documentation](#).

2 shell commands:

- Clone the pipeline repo
- Install dependencies
  - bedtools, biopython, bowti2, jellyfish, numpy, nupack, pandas, pybedtools, sam2pairwise, samtools, sklearn, scipy, xgboost, zip
- Run genome-wide pipeline on test files

# Outline

- Homework 5 wrap-up
- Related topics:
  - JupyterLab IDE on GS cluster
  - GitHub basics
  - Virtual environments and conda
  - Intro to Snakemake
- Homework 6 questions

# Homework 6 Overview

chm13.chr16.txt

Goal: to find CNVs using D-segments

```
16      1      0
16      2      0
16      3      0
16      4      0
16      5      0
      [ ... ]
16     14793    0
16     14794    1
16     14795    3
16     14796    0
      [ ... ]
```

**Data:** next-gen read alignments to genome, CHM13 chr16

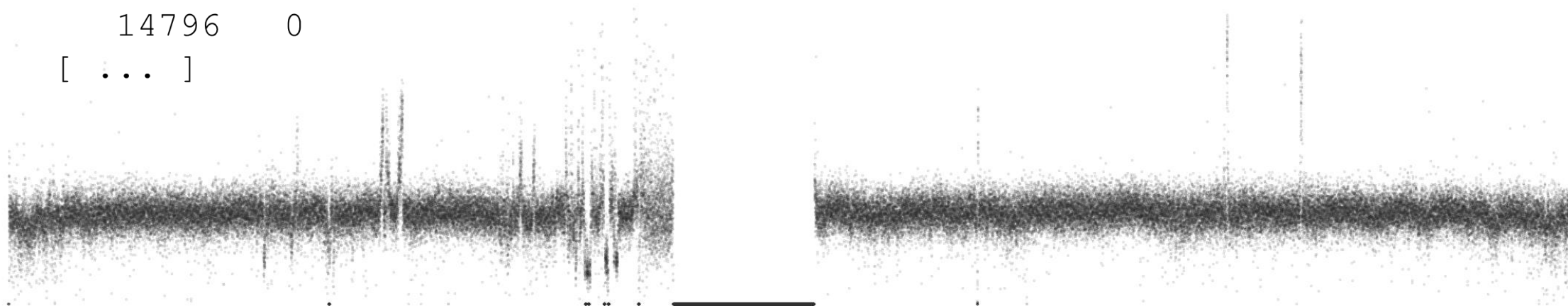
**Observed symbols:** counts of read starts at each position

- Frequencies from Poisson dist. with appropriate mean

**Target regions:** heterozygous duplications

- One chrom = ref allele, other = dup, Poisson mean 1.5X background

Avg. #  
Reads



Position (chr16)



# Homework 6 Overview

chm13.chr16.txt

```
16      1      0
16      2      0
16      3      0
16      4      0
16      5      0
      [ ... ]
16     14793    0
16     14794    1
16     14795    3
16     14796    0
      [ ... ]
```

D-segment algorithm

$O(N)$  algorithm to find all maximal D-segs:

```
cumul = max = 0; start = 1;
```

```
for (i = 1; i ≤ N; i++) {
```

```
    cumul += s[i];
```

```
    if (cumul ≥ max)
```

```
        {max = cumul; end = i;}
```

```
    if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
```

```
        if (max ≥ S)
```

```
            {print start, end, max; }
```

```
            max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING  
                NEEDED! */
```

```
    }
```

```
}
```

# Homework 6 Overview

## Annotations for top 3 scoring segments

### D-segment info

Assignment: GS 540 HW6  
Name: Conor Camplisson  
Email: concamp@uw.edu  
Language: Python  
Runtime: 1.058 sec

Segment Histogram:  
Non-Elevated CN Segments=8  
Elevated CN Segments=7

Segment List:  
48164 48273 66.76  
67646 68115 97.51  
105528 106003 63.04  
106904 107345 41.67  
122792 123034 66.56  
164376 164665 62.09  
165086 166103 225.95

Annotations:

Start: 165086  
End: 166103  
Description: Something interesting (e.g., "Overlaps with exon5 of the protein coding gene cMyc")

Start: 67646  
End: 68115  
Description: Something interesting (e.g., "Overlaps with exon5 of the protein coding gene cMyc")

Start: 48164  
End: 48273  
Description: Something interesting (e.g., "Overlaps with exon5 of the protein coding gene cMyc")

### Read start count histograms

Read start histogram for non-elevated copy-number segments:

0=331908  
1=19439  
2=4272  
>=3=1332

Read start histogram for elevated copy-number segments:

0=1656  
1=542  
2=352  
>=3=499

# Reminders

- Homework 6 due this Sunday Feb. 19, 11:59 pm
- Homework 7 will be posted tomorrow

