AMR Genomics: Reads to Reports

Finlay Maguire (Dalhousie University)

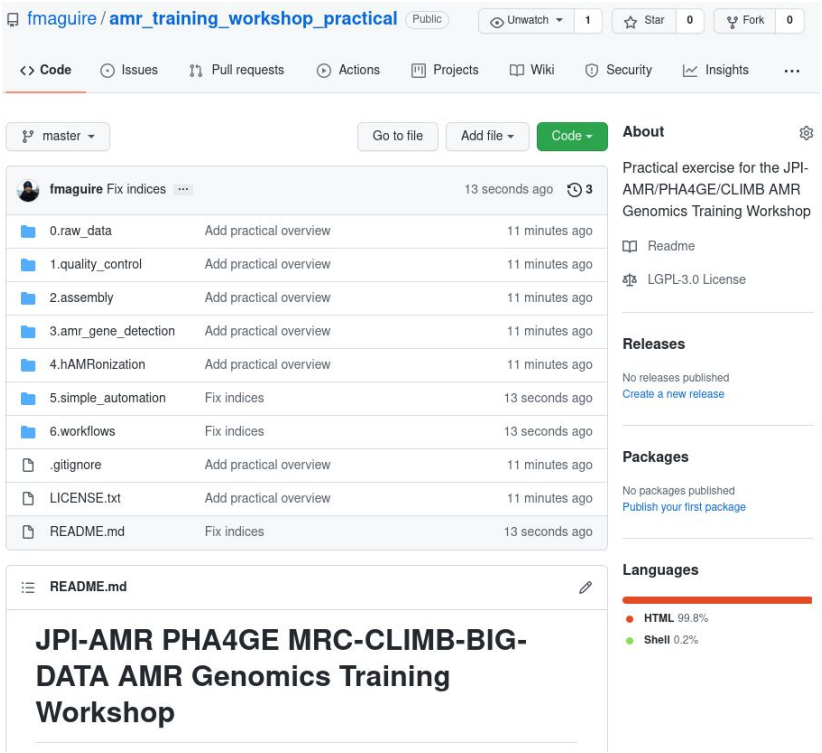
Overview

- Quality control
- Trimming/Error Correction
- Genome Assembly
- Predicting AMR Genes from Assemblies
- Standardising Output
- Workflows

Things that won’t be covered:

- Read-based analyses
- Metagenomics
- Many alternative tools!

Materials



github.com/fmaguire/amr_training_workshop_practical

Setting Yourself Up

- Bioinformatics relies heavily on the UNIX shell:
  - Linux
  - Mac's OSX
  - Windows Subsystem Linux (WSL)
- Package managers and containers make installation easier
- Environments prevent tools getting in each other's way
- Bioconda provides both (docs.conda.io/en/latest/miniconda.html + bioconda.github.io)

```
conda create -n amr fastp shovill ncbi-amrfinderplus hAMRonization
conda activate amr
```

Garbage In - Garbage Out: Quality Control

- Positive and Negative Controls
- Contamination checks (e.g., kraken2 + krona)
- Sequencing quality checks
  - Quality scores
  - Over-representation
  - N’s

Tidying Up Your Reads: Trimming/Error Correction

```
fastp --in1 sa
      --out1 s
```

Before filtering

total reads:	5.512546 M
total bases:	689.068250 M
Q20 bases:	664.553262 M (96.442299%)
Q30 bases:	641.367221 M (93.077459%)
GC content:	37.854518%

After filtering

Filtering result

reads passed filters:	5.403942 M (98.029876%)
reads with low quality:	105.014000 K (1.905000%)
reads with too many N:	3.590000 K (0.065124%)
reads too short:	0 (0.000000%)

```
_trimmed.fq.gz
```

Adapters

Adapter or bad ligation of read1

The input has little adapter percentage (~0.007316%), probably it's trimmed before.

Sequence	Occurrences
all adapter sequences	627

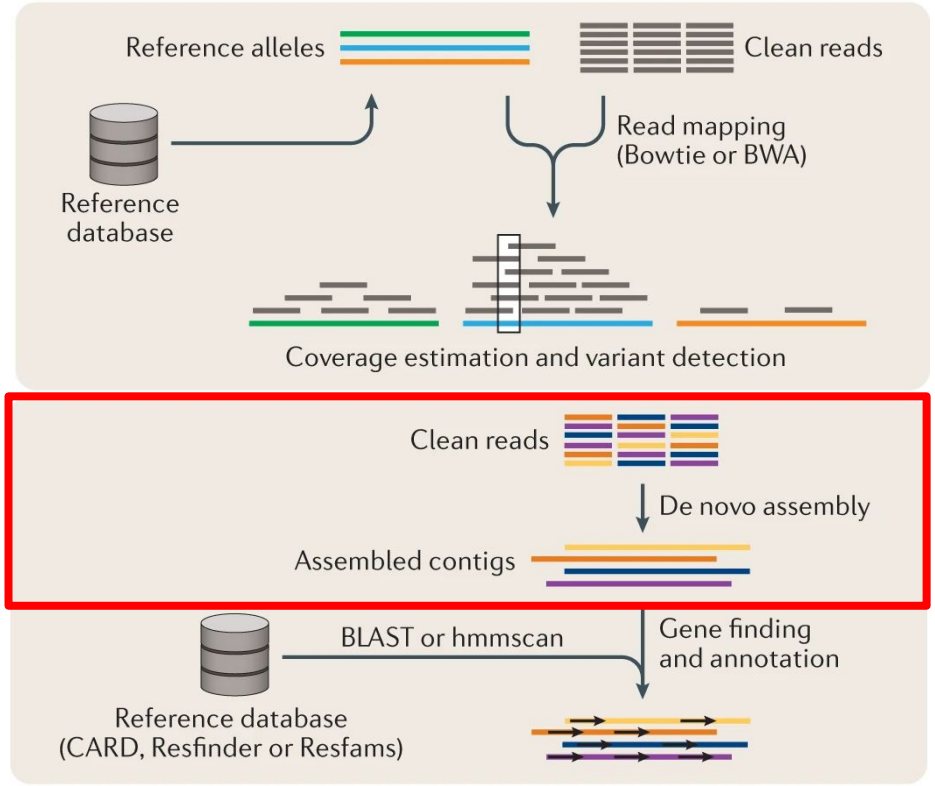
Adapter or bad ligation of read2

The input has little adapter percentage (~0.007316%), probably it's trimmed before.

Sequence	Occurrences
all adapter sequences	627

Note: reads in repository are simulated to be simple; trimming isn't doing much to them

Turning Reads Into a Genome



10.1038/s41576-019-0108-4

De novo Assembly

Assembly theory **vs** **Practical assembly**

Similar patterns. Millions of pieces. Missing pieces. Damaged pieces.
And you don't know the right answer!

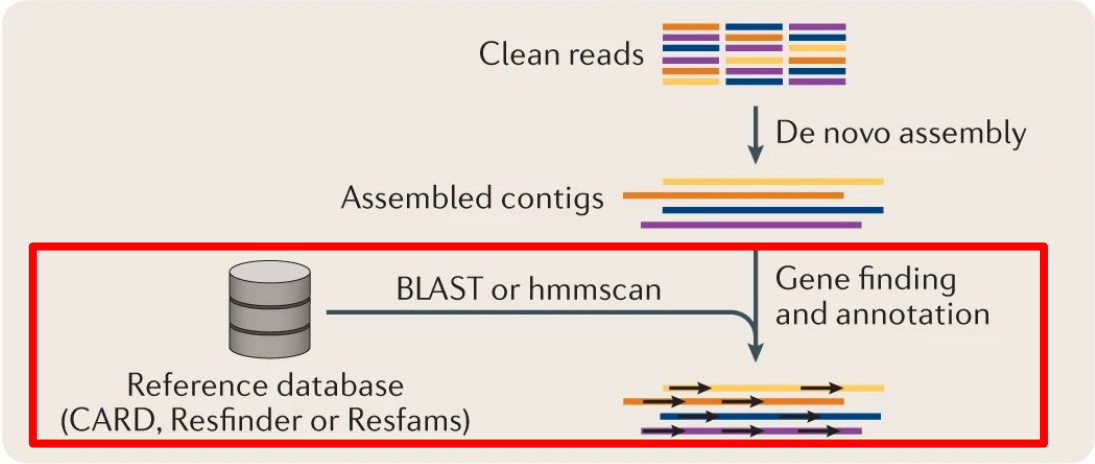
10.1007/s12575-009-9004-1
twitter.com/lh3lh3/status/1448772264300847104

Shovill

- Estimate genome size (mash)
- Downsample reads (seqtk) to ~100x
- Trim reads (trimmomatic)
- Error correct reads (lighter)
- Stitch overlapping reads (Flash)
- Assemble reads (SPAdes)
- Correct mistakes (BWA-MEM + Pilon)

```
shovill --R1 sampleA_R1_trimmed.fq.gz --R2 sampleA_R2_trimmed.fq.gz \
        --outdir sampleA_assembly
```

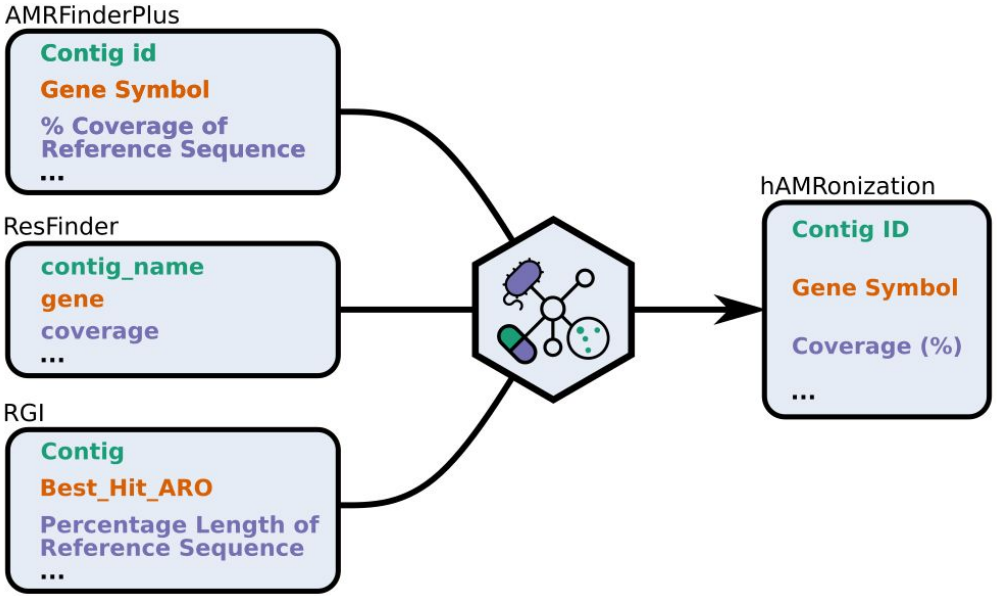
Contig-based AMR Gene Prediction



10.1038/s41576-019-0108-4

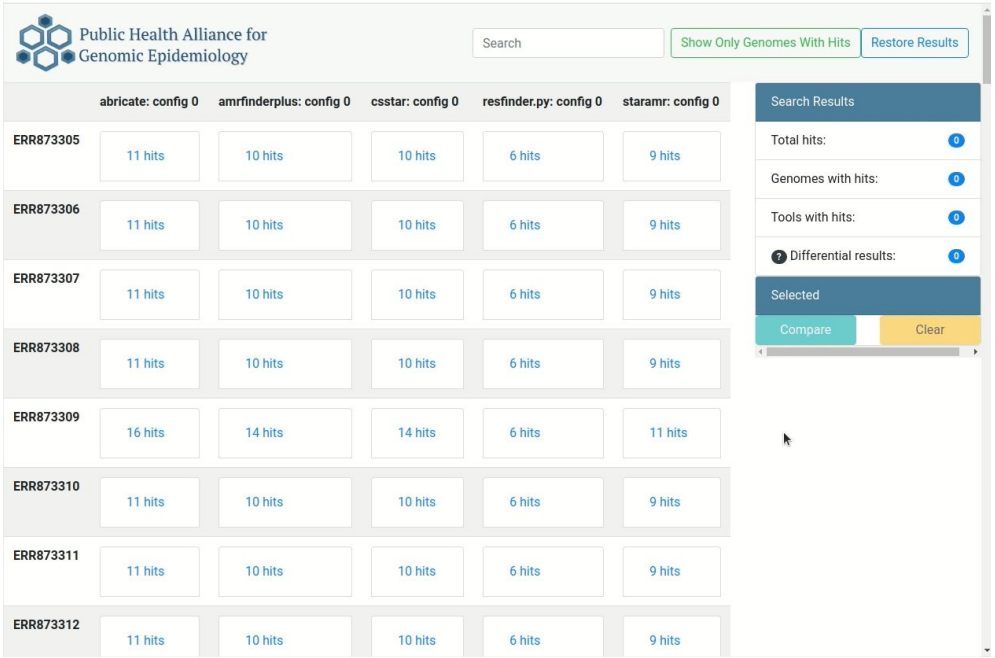
```
amrfinder --update
> Database version: 2021-09-30.1

amrfinder --nucleotide sampleA_assembly/contigs.fa \
          --output amrfinderplus_results.tsv
```

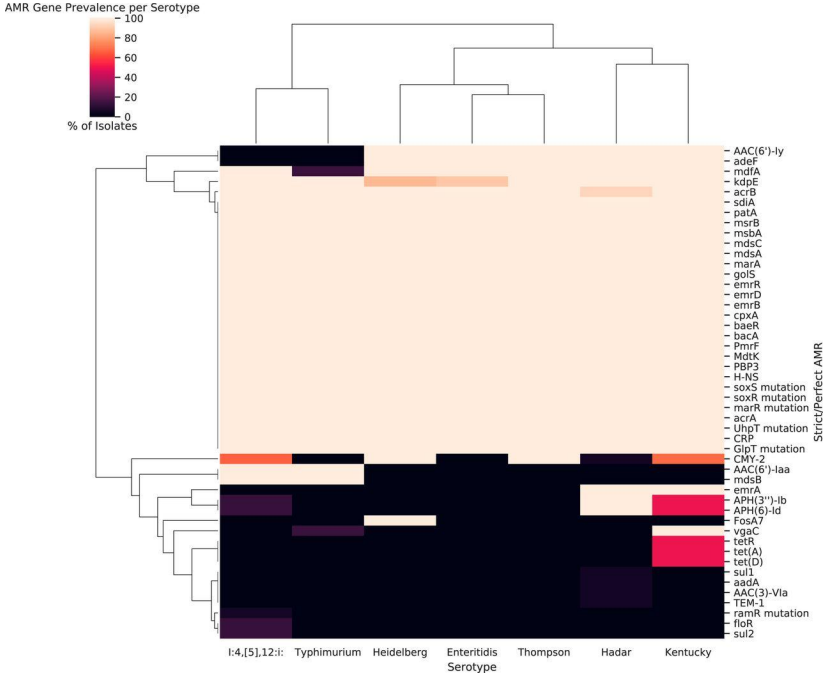
Standardising AMR Detection Results




```
hamronize amrfinderplus --analysis_software_version 3.10.16 \
        --reference_database_version 2021-09-30.1 \
        --input_file_name sampleA amrfinderplus_results.tsv \
        > hAMRonized_amr_report.tsv
```

Summarising Results






```
hamronize summarize --summary_type interactive hAMRonized_amr_report.tsv \
> amr_summary.html
```

Great, that is how we do it for 1 sample but what about 10-1,000s?

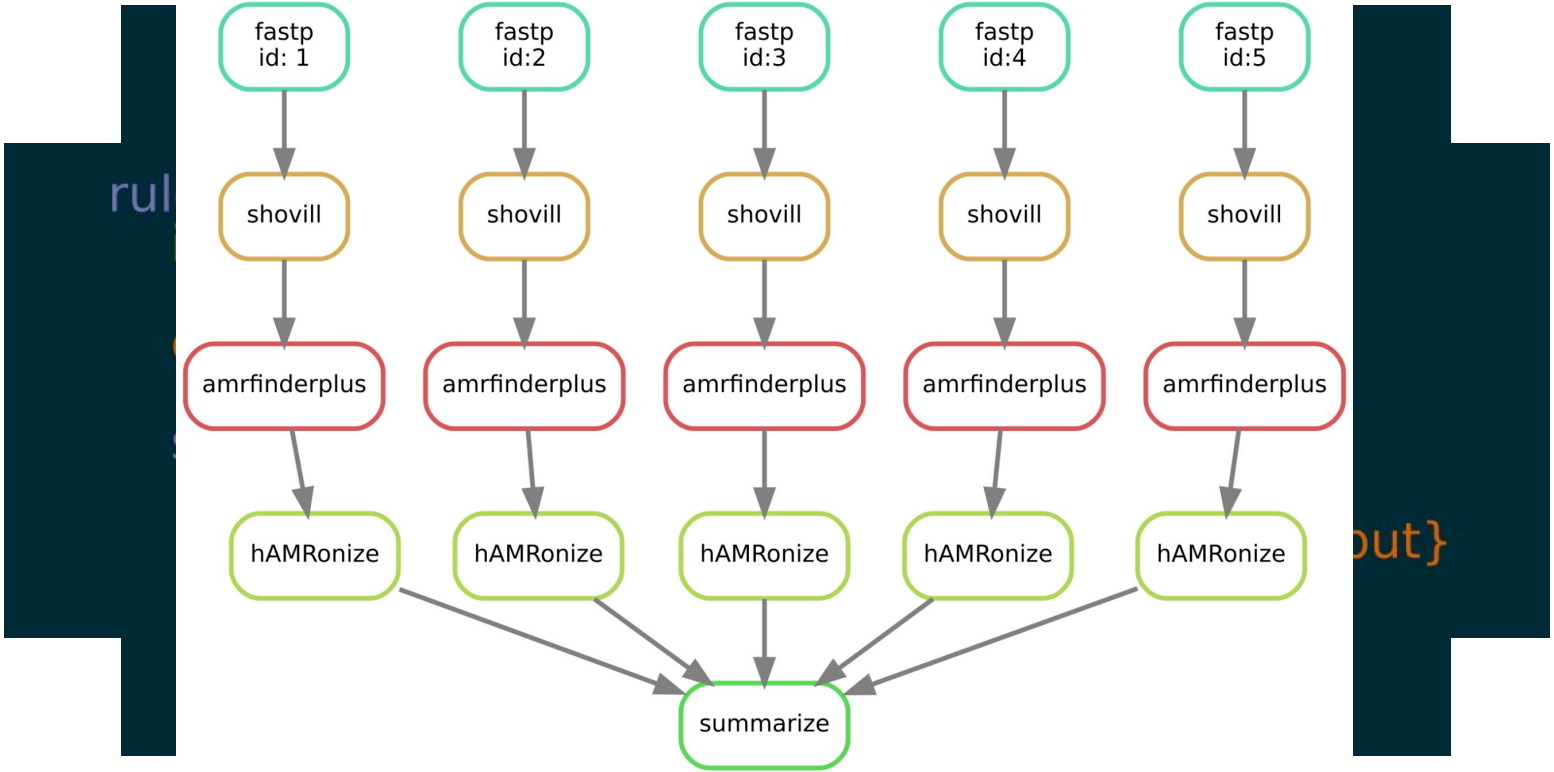
Option 1: Lots of Typing and Mistakes

```
fastp raw_reads_1 > trimmed_reads_1
shovill trimmed_reads_1 > assembly_1
amrfinderplus assembly_1 > amr_predict_1
hamronize amr_predict_1 > hamronized_1

fastp raw_reads_2 > trimmed_reads_2
shovill trimmed_reads_2 > assembly_2
amrfinderplus assembly_2 > amr_predict_2
hamronize amr_predict_2 > hamronized_2

fastp raw_reads_3 > trimmed_reads_3
shovill trimmed_reads_3 > assembly_3
amrfinderplus assembly_3 > amr_predict_3
hamronize amr_predict_3 > hamronized_3
```

Option 2: Bash Loop

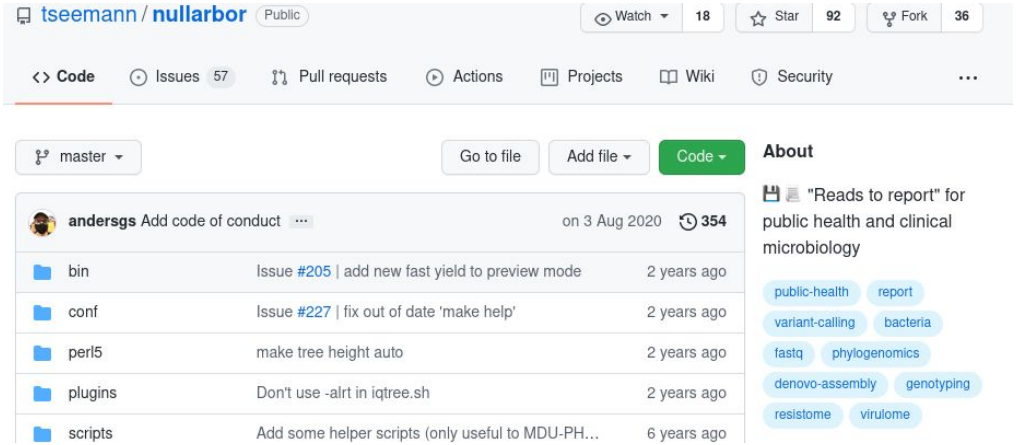
```
        fastp raw_reads_1 > trimmed_reads_1
        shovill trimmed_reads_1 > assembly_1
        amrfinderplus assembly_1 > amr_predict_1
        hamronize amr_predict_1 > hamronized_1

for sample in $(seq 1 10000); do
    fastp raw_reads_$sample > trimmed_reads_$sample
    shovill trimmed_reads_$sample > assembly_$sample
    amrfinderplus assembly_$sample > amr_predict_$sample
    hamronize amr_predict_$sample > hamronized_$sample
done

        shovill trimmed_reads_3 > assembly_3
        amrfinderplus assembly_3 > amr_predict_3
        hamronize amr_predict_3 > hamronized_3
```

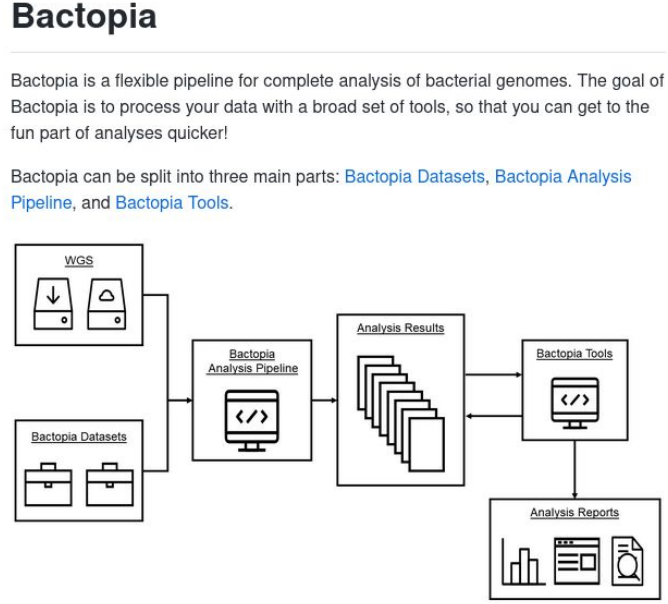
Option 3: Workflows!



Option 4: Someone else's workflow!

- Galaxy (user friendly and not terminal-based): Published Community Workflows (use highly rated/used)
- Snakemake (familiar python but difficult model): "Snakemake workflows" (curated best practice workflows).
- Nextflow (unfamiliar language but simple model): "nf-core" (curated best practice workflows)





Take-aways

- Bioinformatics is built around terminals
- Use conda environments or containers to install tools
- Always read the documentation and help messages for bioinformatics tools
- Robust quality control and controls are vital
- Consider hAMRonizing your AMR gene predictions
- Workflows let you efficiently and robustly run an analysis over many samples
- Don’t reinvent the wheel: use high quality workflows that already exist (modify if you have to)

Acknowledgements

- Our speakers:
  - Dr. Kara Tsang (LSHTM)
  - Dr. Mike Feldgarden (NCBI/NIH)
  - Inês Mendes (IMM/ULisboa)
- Organising committee:
  - Dr. Finlay Maguire, Dr. Andrew Page, Dr. Emma Griffiths, Prof. Mark Pallen, **Lisa Marchioretto**, Dr. Jessica Boname, Dr. Carolyn Johnson
  - JPI-AMR, PHA4GE, MRC-CLIMB-BIG-DATA
- Panelists:
  - A/Prof. Henk den Bakker (UGeorgia)
  - Dr. Anthony Underwood (CPGS)