Session 13 - Hands-on Practical: Running nf-core/funcscan on Bacterial Genome Assemblies

Objective

This tutorial walks you through running the nf-core/funcscan pipeline on locally assembled bacterial genomes using various container technologies. You will:

- Execute functional screening modules: AMP, ARG, and BGC

- Use Singularity, Docker, Podman, and Conda

- Apply custom resource configuration for low-memory environments

1. Overview of nf-core/funcscan

nf-core/funcscan is a modular Nextflow DSL2 pipeline that screens nucleotide FASTA files for:

- Antimicrobial peptides (AMPs)

- Antibiotic resistance genes (ARGs)

- Biosynthetic gene clusters (BGCs)

It ensures reproducibility, scalability, and containerized environments (Docker, Singularity, etc.).

2. ⚙️ Prerequisites

Ensure the following:

- Linux system with Docker, Singularity, Podman or Conda

- Nextflow ≥ 23.10.0

- FASTA files of bacterial assemblies

- Custom config to handle limited RAM systems

3. 📂 Dataset

Assembly directory: $HOME/module5/assemblies/

Sample sheet (samples.csv):

```
sample,fasta
bc01,/home/nguinkal/module5/assemblies/bc01.fasta
bc02,/home/nguinkal/module5/assemblies/bc02.fasta
bc03,/home/nguinkal/module5/assemblies/bc03.fasta
bc04,/home/nguinkal/module5/assemblies/bc04.fasta
```

4. 🧾 Custom Resource Configuration (base.config)

Save this as base.config to override memory defaults and avoid antiSMASH failures:

```
process {
  memory = '12 GB'
  cpus = 4
  time = '6h'

  withName: 'NFCORE_FUNCSCAN:FUNCSCAN:BGC:ANTISMASH_ANTISMASHLITE' {
    memory = '12 GB'
    cpus = 8
    time = '12h'
  }
}
```

5. Running the Pipeline (Per Container Profile)

5.1 Singularity

```
nextflow run nf-core/funcscan \
  -profile singularity \
  --input samples.csv \
  --outdir funscan_out_singularity \
  --run_amp_screening \
  --run_arg_screening \
  --run_bgc_screening \
  -c base.config
```

5.2 Docker

```
nextflow run nf-core/funcscan \
  -profile docker \
  --input samples.csv \
  --outdir funscan_out_docker \
  --run_amp_screening \
  --run_arg_screening \
  --run_bgc_screening \
  -c base.config
```

5.3 Conda

```
nextflow run nf-core/funcscan \
  -profile conda \
  --input samples.csv \
  --outdir funscan_out_conda \
  --run_amp_screening \
  --run_arg_screening \
  --run_bgc_screening \
  -c base.config
```

5.4 Podman

```
nextflow run nf-core/funcscan \
  -profile podman \
  --input samples.csv \
  --outdir funscan_out_podman \
  --run_amp_screening \
  --run_arg_screening \
  --run_bgc_screening \
  -c base.config
```

6. 📤 Output Structure

Each run generates:

- results/: AMP, ARG, BGC outputs

- multiqc/: Quality control reports

- work/: Intermediate files (can be cleaned after success)

7. 🛠️ Troubleshooting Tips

- Inspect .nextflow.log for errors

- Adjust memory with withName: blocks in config

- Ensure FASTA paths are absolute and readable

- Use -resume to skip completed steps on re-runs

8. 📈 Next Steps

- Benchmark runtime/memory across profiles (see *benchmark_funscan.sh* script)

- Compare screening results across samples/profiles

🔗 Useful Links

- https://nf-co.re/funcscan

- https://github.com/nf-core/funcscan

- https://www.nextflow.io/docs/latest/index.html

9. 📊 Summary Table for run profiles

Profile	Container Tech	Output Directory	Memory Override	Notes
Singularity	Singularity	funscan_out_singularity	Yes (base.config)	Ideal for HPC, no root access needed
Docker	Docker	funscan_out_docker	Yes	Needs Docker daemon (sudo/root)
Conda	Conda	funscan_out_conda	Yes	Use where containers are not allowed
Podman	Podman	funscan_out_podman	Yes	Rootless container alternative