The entire code repository can be found at `https://github.com/dgsaf/hpc-assignment-4`.

Contents

List of Figures

List of Tables

1 Interpretation

The `count` process is presented in Listing 4.

```
process count {
  input:
  path(files) from files_ch.collect()

  output:
  file("results.csv") into counted_ch

  container = ""

  shell:
  '''
  echo "seed,ncores,nsrc" > results.csv
  files=($(ls table*.csv))
  for f in ${files[@]}; do
    seed_cores=($(echo ${f} | tr '_.' ' ' | awk '{print $2 " " $3}'))
    seed=${seed_cores[0]}
    cores=${seed_cores[1]}
    nsrc=$(echo "$(cat ${f} | wc -l) - 1" | bc -l)
    echo "${seed},${cores},${nsrc}" >> results.csv
  done
  '''
}
```

Listing 1: The process `process count` *in* `nextflow/main.nf`*.*

i. `echo "seed,ncores,nsrc" > results.csv`

   The output of the left hand side (SNR seed, number of cores used and number of sources counted) which would normally be sent to `stdout` is redirected to the destination specified on the right hand side (in this case, a file called `results.csv`) creating/overwriting the file in the process.

ii. `echo "${seed},${cores},${nsrc}" >> results.csv`

   Similar to the effect of `>` above, except that this operator appends to the file, rather than overwriting it.

iii. `cat ${f} | wc -l`

   The output of the left hand (normally send to `stdout`) is *piped* to input of the right hand side; that is, the input of `wc -l` is taken from the output of `cat ${f}`. This effect of this command is to count the number of lines in the file with filepath `${f}`.

iv. `$(<command>)` and `($(<command>))`

   Wrapping a shell command with a single pair of parentheses, `(<command>)` executes that command in a sub-shell - prefixing this with a dollar sign captures the final output of that subshell `$(<command>)`. However if the result of this is a list of whitespace-delimited strings, wrapping this in a further set of parentheses, `($(<command>))`, will capture the output as an array variable.

v. `$(ls table*.csv)`

The effect of `ls table*.csv` is to list all files in the current directory which start with `table` and end with `.csv`. The effect of `$(ls table*.csv)` is to capture this list of matching files as a string. Specifically, it collects the files containing tables of counted sources for all combinations of `seed` and `cores`.

vi. `echo ${f}`

The effect of this command is to output (to `stdout`) the string-value of the variable `f`. Specifically, this outputs the filepath of a specific file, which is taken from the list of tables described above, and will be of the form `table_<seeds>_<cores>.csv`.

vii. `tr '_.' ' '`

The effect of this command is, for a given input, to translate all instances of the characters `_` and `.` into whitespace. Specifically this transforms `table_<seeds>_<cores>.csv` into `table <seeds> <cores> csv`.

viii. `awk '{print $2 " " $3}'`

The effect of this command is, for a given input consisting of lines (delimited by `\n`) containing records (delimited by whitespace), to print the second record, followed by a white space, followed by the third record, for each line in the input. Specifically, this transforms `table <seeds> <cores> csv` into `<seeds> <cores>`.

ix. `cat ${f}`

x. `wc -l`

xi. `echo "$(cat ${f} | wc -l)-1" | bc -l`

2 Development

```
counted_ch.into{counted_for_ch; counted_xargs_ch}
```

Listing 2: The channel `counted_ch` *is duplicated, with one for each of* `process plot_for`*, and* `process plot_xargs`*, in* `nextflow/main.nf`*.*

2.1 Bash For Loop

```
process plot_for {
  input:
  path(table) from counted_for_ch

  output:
  file("*.png") into final_for_ch

  cpus 4

  shell:
  '''
  ncores_set=$(awk -F, '{if (NR != 1) {print $2}}' !{table} | sort | uniq)

  for ncores in $ncores_set ; do
    python !{projectDir}/plot_completeness.py \
    --infile !{table} --outfile plot_for_${ncores}.png --cores $ncores
  done
  '''
}
```

Listing 3: The process `process plot_for` *in* `nextflow/main.nf`*.*

2.2 xargs Command

```
process plot_xargs {
  input:
  path(table) from counted_xargs_ch

  output:
  file("*.png") into final_xargs_ch

  cpus 4

  shell:
  '''
  ncores_set=$(awk -F, '{if (NR != 1) {print $2}}' !{table} | sort | uniq)

  printf '%s ' $ncores_set | xargs -n1 -P4 -I ncores -d ' ' \
  python !{projectDir}/plot_completeness.py \
  --infile !{table} --outfile plot_xargs_ncores.png --cores ncores
  '''
}
```

Listing 4: The process `process plot_for` *in* `nextflow/main.nf`*.*

3 Execution

3.1 SNR Plot

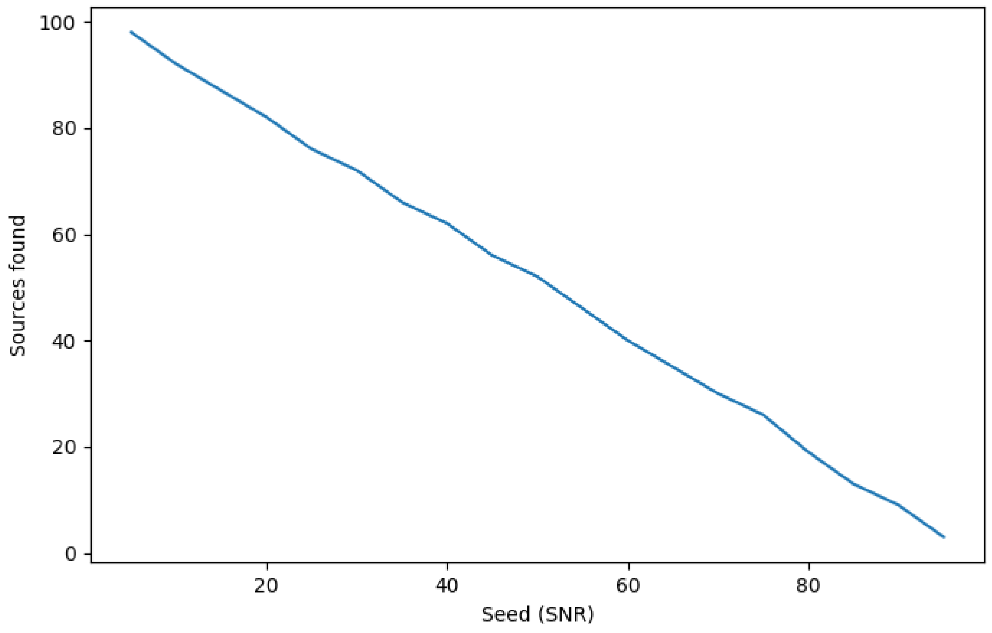

Figure 1: The Signal-to-Noise Ratio (SNR) is presented for a range of seed SNR values. This figure was produced by `process plot_for`*, for the case of* `cores = 1`*. No difference was observed between this plot and any of the other plots produced for any value of* `cores`*, nor whether if* `process plot_for` *or if* `process plot_xargs` *were used.*

3.2 Workflow DAG

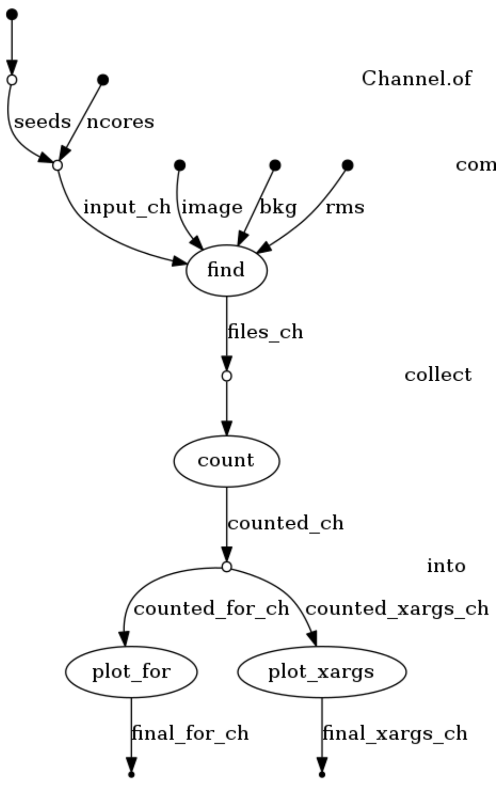

Figure 2: The Directed Acyclic Graph (DAG) of the workflow is presented. Note that the `combine` *operator (below* `Channel.of`*) appears to have been cropped out by the tool producing the DAG.*

4 Analysis