MAKE assignment

Jakub Widawski, NIU:1551069

2/3/2020

TASKS

Using the make utility, create a makefile that generates index references for BWA mapper. Your makefile should take references in fasta format from a references folder. For each reference (ref_x.fasta or ref_x.fa) you should generate the corresponding ref_x.pac, ref_x.amb, ref_x.ann, ref_xbtw and ref_x.sa files. All these files should be in a folder named index/ref_x . See http://bio-bwa.sourceforge.net/ for details about BWA index generation.

Add batch processing support in your previous makefile (i.e. generation of indexes should be done by submitting jobs to SLURM).

Your solution should include a brief report describing your solution as well as the corresponding make and submit files . All this information should be compressed into a single file named MAKE-.tar

Don't forget to add also your name at the brief report.

Makefile1, no slurm support:

```
## First we define the input and index (indexed) dir
INPUTDIR=./references/
INDEXDIR=./index/

## We now find the relative path to all the .fa and .fasta files
INPUTS_FA=$(wildcard $(INPUTDIR)*.fa)
INPUTS_FASTA=$(wildcard $(INPUTDIR)*.fasta)

## Compute input names without the extensions
NAMES_FA=$(patsubst $(INPUTDIR)%.fa, %, $(INPUTS_FA))
NAMES_FASTA=$(patsubst $(INPUTDIR)%.fasta, %, $(INPUTS_FASTA))

## Get the target directories we want to make
## Their names are based on the basename of the input files
TARGETDIRS_FA=$(patsubst $(INPUTDIR)%.fa, $(INDEXDIR)%/, $(INPUTS_FA))
TARGETDIRS_FASTA=$(patsubst $(INPUTDIR)%.fasta, $(INDEXDIR)%/, $(INPUTS_FASTA))

## Create variable NAMES containing names of both .fa and .fasta files
NAMES=$(NAMES_FA) $(NAMES_FASTA)

## Create variable TARGETDIRS containing target directories for our output
TARGETDIRS=$(TARGETDIRS_FA) $(TARGETDIRS_FASTA)

## Get names of one for each of our targets
## (One of the BWA index output files has extension .ann) as a variable TARGETNAMES
TARGETNAMES=$(addsuffix .ann, $(NAMES))

 ## Use phony to avoid duplicate name problems
.PHONY: all clean
all: $(TARGETNAMES) ## all is dependent on our array of target names
```

```
## Our target directories do not have dependencies
$(TARGETDIRS):
  ## the mkdir function is called during completion of each of the %.ann targets
    @mkdir -p $@

## Call ./index/{input file} (from $(TARGETDIRS)) to make the directory
%.ann: $(INDEXDIR)%/
    @echo ""
    @echo Indexing file: $* ## Echo the name of the file
    @echo ""

    ## If input file == {file_name}.fa then:
    @if test -f $(INPUTDIR)$*.fa; then bwa index \
    -p $(INDEXDIR)$*/$* $(INPUTDIR)$*.fa; fi

    ## If input file == {file_name}.fa then:
    @if test -f $(INPUTDIR)$*.fasta; then bwa index \
    -p $(INDEXDIR)$*/$* $(INPUTDIR)$*.fasta; fi

## Clean directory only needs to recursively delete
## the index directory with the output files
clean:
    rm -r index
```

Makefile2, with slurm support:

```
## First we define the input and index (indexed) dir
INPUTDIR=./references/
INDEXDIR=./index/

## We now find the relative path to all the .fa and .fasta files
INPUTS_FA=$(wildcard $(INPUTDIR)*.fa)
INPUTS_FASTA=$(wildcard $(INPUTDIR)*.fasta)

## Compute input names without the extensions
NAMES_FA=$(patsubst $(INPUTDIR)%.fa, %, $(INPUTS_FA))
NAMES_FASTA=$(patsubst $(INPUTDIR)%.fasta, %, $(INPUTS_FASTA))

## Get the target directories we want to make
## Their names are based on the basename of the input files
TARGETDIRS_FA=$(patsubst $(INPUTDIR)%.fa, $(INDEXDIR)%/, $(INPUTS_FA))
TARGETDIRS_FASTA=$(patsubst $(INPUTDIR)%.fasta, $(INDEXDIR)%/, $(INPUTS_FASTA))

## Create variable NAMES containing names of both .fa and .fasta files
NAMES=$(NAMES_FA) $(NAMES_FASTA)

## Create variable TARGETDIRS containing target directories for our output
TARGETDIRS=$(TARGETDIRS_FA) $(TARGETDIRS_FASTA)

## Get names of one for each of our targets
## (One of the BWA index output files has extension .ann) as a variable TARGETNAMES
TARGETNAMES=$(addsuffix .ann, $(NAMES))
```

```
 ## Use phony to avoid duplicate name problems
.PHONY: all clean
all: $(TARGETNAMES) ## all is dependent on our array of target names

## Our target directories do not have dependencies
$(TARGETDIRS):
  ## the mkdir function is called during completion of each of the %.ann targets
    @mkdir -p $@

## Call ./index/{input file} (from $(TARGETDIRS)) to make the directory
%.ann: $(INDEXDIR)%/
    @echo ""
    @echo Indexing file: $* ## Echo the name of the file
    @echo ""

    ## If input file == {file_name}.fa then:
    @if test -f $(INPUTDIR)$*.fa; then sbatch --partition=research.q \
  --output=$*.out --wrap="bwa index -p $(INDEXDIR)$*/$* $(INPUTDIR)$*.fa"; fi

    ## If input file == {file_name}.fa then:
    @if test -f $(INPUTDIR)$*.fasta; then sbatch --partition=research.q \
    --output=$*.out --wrap="bwa index -p $(INDEXDIR)$*/$* $(INPUTDIR)$*.fasta"; fi

## Clean directory only needs to recursively delete
## the index directory with the output files
clean:
    rm -r index
```

Brief description of the solution

First we define the input and index (indexed) dir using the wildcard function. We then compute arrays containing the relative input paths, input file names, target input directory names and input file names with no extension and target names (using one bwa index output extension ==> .ann) for .fa and .fasta files respectively. The "all" target from make then calls for the creation of each of the %.ann targets. The defined dependency calls for the creation of the directory for the output files (./index/{file_name}).

After creation of the output directory, ach of the targets is created with bwa index in the previously initiated output directory (with the -p option from bwa index). This is done using the file name without extension, so we use "test -f" to check if the file existing in the input directory is of extension .fa or .fasta.

With using ".PHONY" we make sure that any conflicts with the names "clean" and "all" are ignored and the make file executes the functions properly

clean will recursively remove the output files from the directory. If we want to also delete the output files created with slurm we can add the line : "-rm *.out" in order to make sure that the directory is back to its initial state. The "-" before the rm command is used to suppress the errors, that is to stop an error being printed out if the .out files were not created.

For the slurm support, we run the sbatch with "–wrap" option to perform the bwa indexing. Each file will have a different output file consisting of bwa index stdout informmation and named like: {file name}.out.