**Cobretti Quick Start Guide:**

sbatch template:

#!/bin/bash -l

#SBATCH --partition=whatever

#SBATCH --nodelist=biocrunch9

#SBATCH --time=1-00:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=1

#SBATCH --job-name=cobretti

#SBATCH --mail-user=[email]@iastate.edu

#SBATCH --mail-type=ALL

module load py-biopython/1.70-py3-wos466g

module load python/3.6.5-fwk5uaj

python /work/LAS/wmoss-lab/scripts/cobretti.py -stage [#] -email [email]@iastate.edu

srun template:

python /work/LAS/wmoss-lab/scripts/cobretti.py -stage [#] -email [email]@iastate.edu

Required arguments:

-stage options: 1A, 1AA, 1AB, 1B, 1BA, 1BB, 1BC, 1C, 1CA, 2A, 2AA, 2AB, 2B, 2BA, 3A, 3B

-email is required, as it will be inserted into shell scripts

Optional requirements:

-i list of fasta accession numbers (used in place of fasta files in stage 1A)

-seq sequence directory (will generate ./sequences directory if not specified)

-db BLAST database directory (will generate ./databases directory if not specified)

-dbn motif directory (will generate ./motifs directory if not specified)

-pk pseudoknot motif directory (will generate ./pk\_motifs directory if not specified)

To change program/script locations:

-tools location of cobrettitools.py

-sf location of ScanFold.py

-cmb location of cm-builder

-rs location of R-Scape

-perl location of Perl (for R-Scape)

-rf location of RNAFramework (for R-Scape)

-ky location of Knotty

-hf location of Iterative HFold

-sim location of SimRNA

-qrnas location of QRNAS

-fpocket location of fpocket

**Stage 1A: ScanFold and BLAST automation**

1. Place fasta files of interest into a folder or create a list of accession numbers (-i).

Make sure fasta files are named the same in both the filename and the header line (“>”). Cobretti will use everything in the header line up to the first underscore, so it is recommended to remove all underscores from both the filename and header before running.

1. srun or sbatch stage 1A.

Will search current working directory for fasta files (.fa or .fasta), move them to ./sequences directory, and then perform ScanFold and BLAST on each sequence. ScanFold results are stored in files according to the names of the fasta files, while BLAST databases are stored in ./databases.

ScanFold settings: --global\_refold.

BLAST settings: blastn, nt database, 2500 target sequences, -max\_hsps 1 (top hit only), will only output aligned portion of sequences to avoid excessive memory allocation during cm-builder runs.

1. Once all scanfold\_[X].sh and blast\_[X].sh jobs are complete, verify that there were no errors: all ScanFold runs properly completed, all BLAST databases created, no failures in the .out logs

Substage options:

1AA ScanFold only

1AB BLAST only

**Stage 1B: Pseudoknot prediction, breakdown, cm-builder run**

1. Add #SBATCH --mem=40G to the shell script. srun or sbatch stage 1B.

Will check for ./sequences, ./ databases, and ./motifs directories, then will clean up any files leftover from ScanFold and BLAST runs.

Looks in ./motifs directory for all .dbn files, extends them by 30 nts on 5’ and 3’ end using sequences in ./sequences directory, creating extended.dbn file. Will then run this list through Knotty, HFold without constraints, and HFold with ScanFold constraints, outputting the results to tmppk.txt. This file will be scrubbed of unnecessary lines, resulting in pkclean.txt. All motifs in pkclean.txt will be checked for pseudoknots and broken down into non-nested pairs. All motifs will then be shortened to remove unpaired 5’ and 3’ nts, and the final results will be output to ./pk\_motifs.

Looks in ./pk\_motifs directory for .dbn files and builds and runs cm-builder scripts. Due to out of memory issues, each script is limited to 10 motifs and any BLAST database over 8GB is run as its own shell script.

1. Once all cmbuilder[X].sh jobs are complete, verify that there were no errors: all motifs extended proiperly (extended.dbn), all pseudoknots folded properly (check tmppk.txt and .out for errors, as an error in Knotty/HFold may still provide a structure but may/may not output it), pseudoknot motifs cleaned up properly (pkclean.txt, should have 4x as many motifs at this point), all pseudoknot motifs created properly (./pk\_motifs), no failures in the .out logs (out of memory or Bad address are both memory issues, either split the shell script into smaller runs or increase the memory)

Substage options:

1BA BLAST cleanup only

1BB Pseudoknot fold only (only use this option if there were major issues building the pseudoknot files, as any error caused by Knotty/HFold will reoccur every time. It is typically faster/easier to look through the pseudoknot files and fix the 1-2 errors manually).

1BC Build and run cm-builder scripts only

**Stage 1C: R-Scape, cleanup**

1. srun or sbatch stage 1C.

Moves non-essential files from cm-builder runs into folders, then uses R-Scape on all .stockholm files in the current working directory. The results are compiled into a single Rscape.pdf file.

R-Scape settings: -s (two-set statistical test), --ntree 10 (10 FastTree alignments, improves E-value repeatability)

Reads all the .power files and compiles the results into a single file, covariance.txt.

Finishes by moving all generated files into folders (./cm, ./power, ./Stockholm, etc.).

1. Once the rscape.sh job is complete, verify that there were no errors: all .pdf and .power files generated, covariance.txt and Rscape.pdf generated, no errors in the .out file.

Substage options:

1CA Final cleanup step only

**Stage 2A: SimRNA run**

1. Place .dbn files of interest into a folder.
2. srun or sbatch stage 2A.

Builds and runs SimRNA shell scripts for all .dbn files in the current working directory.

SimRNA settings: 10 instances with random seeds, 8 replicas, RMSD clusters at 5.0, 7.0, 10.0 and sequence length/10 Angstroms, outputting top 3 clusters for each RMSD value.

1. Once all simrna[#].sh jobs are complete, verify that there were no errors: all \_AA.pdb cluster files generated, no errors in the .out file.

Substage options:

2AA SimRNA preparation only

2AB Run SimRNA shells only

**Stage 2B: QRNAS run, ARES run, fpocket run**

1. Place \_AA.pdb files of interest into a folder (or run in SimRNA location to run all previous results).
2. srun or sbatch stage 2B.

Runs QRNAS on all .pdb files in the current working directory.

1. Once script is complete, verify that there were no errors: all \_QRNAS.pdb files generated, no errors in the .out file or \_QRNAS.log files.

Substage options:

2BA SimRNA preparation only

2BB ARES run only (WIP)

2BC fpocket run only (WIP)

**Stage 3A: Dock 6 run (WIP)**

1. Place .pdb files of interest into a folder.
2. srun or sbatch stage 3A.

Runs DOCK 6 on all .pdb files in the current working directory.

1. Once all dock6.sh scripts are complete, verify that there were no errors: all DOCK 6 files generated, no errors in the .out file.

**Stage 3B: AnnapuRNA run (WIP)**

1. Place .pdb files of interest into a folder.
2. srun or sbatch stage 3B.

Runs Annapurna on all .pdb files in the current working directory.

1. Once script is complete, verify that there were no errors: all AnnapuRNA files generated, no errors in the .out file.