**Cobretti User Manual**

Cobretti is a tool for high-throughput automation of several RNA bioinformatic programs. The main purposes are to increase the number of RNA structures that can be characterized, reduce the time required to process, and improve repeatability. The main function of this code is to write and execute SLURM scripts on a high-performance computing node, modifying files to meet program input requirements and providing compiled output results for easier user analysis.

**Cobretti Reference Guide:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Files** | **Location** | **Stage** | **Description** | **Substage** | |
| .fa or .fasta | cwd | **1A** | ScanFold 2.0 | **1AA** | |
| BLAST | **1AB** | |
| .fa or .fasta | ./sequences | **1B** | BLAST cleanup | **1BA** | |
| .dbn | ./motifs | PK extend/fold | **1BB** | |
| \_db.txt | ./databases | cm-builder shells | **1BC** | **1BC1** |
| .dbn | ./pk\_motifs | cm-builder run |  |
| .stockholm | cwd | **1C** | cm-builder cleanup |  | |
| R-Scape run |  | |
| R-Scape cleanup | **1CA** | |
| .dbn | cwd | **2A** | SimRNA prep | **2AA** | |
| SimRNA run | **2AB** | |
| \_AA.pdb | cwd | **2B** | SimRNA cleanup | **2BA** | |
| QRNAS prep | **2BB** | |
| QRNAS run | **2BC** | |
| \_QRNAS.pbd | cwd | **2C** | QRNAS cleanup | **2CA** | |
| ARES prep | **2CB** | |
| ARES run | **2CC** | |
| \_QRNAS.pdb | cwd | **2D** | fpocket run |  | |
| .pdb | cwd | **3A\*** | DOCK 6 prep | **3AA\*** | |
| DOCK 6 run | **3AB\*** | |
| .pdb | cwd | **3B\*** | AnnapuRNA run |  | |
|  |  | \* WIP |  |  | |

**Cobretti Install Guide:**

Cobretti itself is a Python script that can be run with Biopython. Simply download the most recent version from GitHub (<https://github.com/moss-lab/Cobretti>) and modify the program locations to their current directories. Current programs used by Cobretti include:

Biopython 1.70 (<https://biopython.org/wiki/Download>)

Python 3.6.5 (<https://www.python.org/downloads/release/python-365/>) (ScanFold requires 3.6.5)

Perl (<https://www.perl.org/get.html>)

ViennaRNA (<https://github.com/ViennaRNA/ViennaRNA>)

ScanFold (<https://github.com/moss-lab/ScanFold>)

Knotty (<https://github.com/HosnaJabbari/Knotty>)

Iterative HFold (<https://github.com/HosnaJabbari/Iterative-HFold>)

RNAFramework (<https://github.com/dincarnato/RNAFramework>)

Infernal (<https://github.com/EddyRivasLab/infernal>)

cm-builder (<https://github.com/dincarnato/labtools>)

R-scape 2.0.0.k (<https://github.com/EddyRivasLab/R-scape>)

SimRNA (<https://genesilico.pl/software/stand-alone/simrna>)

QRNAS (<https://genesilico.pl/software/stand-alone/qrnas>)

ARES (<https://zenodo.org/record/5088971#.Y4Y7rBTMKUk>)

fpocket (<https://github.com/Discngine/fpocket>)

**Cobretti Quick Start Guide:**

**Default shell template:**

Below is the batching template for Cobretti. It is recommended to modify this template (and the shell\_build\_start function within Cobretti) if other settings are desired. Some of these values are further modified by other functions depending on computational/time requirements, and values in square brackets must be provided by the user.

#!/bin/bash -l

#SBATCH --partition=biocrunch

#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

module load python

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

**command line template:**

module load py-biopython

module load python

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

**Required arguments:**

-stage options: 1A, 1AA, 1AB, 1A1, 1A1A, 1B, 1BA, 1BB, 1BC, 1BC1, 1C, 1CA, 2A, 2AA, 2AB, 2B, 2BA, 2BB, 2BC, 2C, 2CA, 2CB, 2CC, 2D, 3A, 3AA, 3AB, 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)

-set\_dbn\_extend

True/False (default = True), used to troubleshoot stage 1B/1BB

**To temporarily change program/script locations:**

-cobretti location of cobretti.py

-sf location of ScanFold.py

-sf2 location of ScanFold2.0.py

-sf2env location of ScanFold2.0 conda environment

-cmb location of cm-builder

-rs location of R-Scape

-r2r location of R2R

-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

-qrnasff location of QRNAS force field directory

-ares location of ARES

-aresenv location of ARES Conda environment

-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 (using -i).

All fasta files will be automatically renamed based on the header line (“>”) of the fasta file. Cobretti will use everything up to the first space (“ “), and will replace dots (“.”) and underscores (“\_”) with dashes (“-“). This is the naming convention that will be used for all produced files.

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 directories named after the header lines, while BLAST results 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 2.0 only

1AB BLAST only

1A1 ScanFold only

1A1A ScanFold only

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

1. Add #SBATCH --mem=40G to the initial 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 the 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 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.

Note: Knotty and Iterative HFold cannot operate on ambiguous nts (N, R, Y, etc.), so Cobretti will trim all ambiguous nts from the 5’ and 3’ ends and replace the rest with a random equivalent (e.g., replace R with A or G).

1. 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.
2. Once all cmbuilder[X].sh jobs are complete, verify that there were no errors: all motifs extended properly (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 (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 fix the 1-2 errors and run this substage than rerun all motifs).

1BC Build and run cm-builder scripts only

1BC1 Build cm-builder scripts, don’t run

**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 .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 (runs automatically after R-Scape run)

**Stage 2A: SimRNA prep and 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**

1. Place all-atom (\_AA.pdb) files of interest into a folder (or run in stage 2A location to run all results).
2. srun or sbatch stage 2B.
3. Cleans up SimRNA results (if any within current directory).
4. Prepares batch scripts and runs QRNAS on all .pdb files in the current working directory.
5. 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 cleanup only

2BB QRNAS prep only

2BC QRNAS run only

**Stage 2C: ARES run**

1. Place all QRNAS (\_QRNAS.pdb) files of interest into a folder (or run in stage 2B location to run all results).
2. srun or sbatch stage 2C.
3. Cleans up QRNAS results (if any within current directory).
4. Prepares batch scripts and runs ARES on all .pdb files in the current working directory.
5. Once script is complete, verify that there were no errors: ARES predictions created, no errors in the .out file.

**Substage options:**

2CA QRNAS cleanup only

2CB ARES prep only

2CC ARES run only

**Stage 2D: fpocket run**

1. Place all QRNAS (\_QRNAS.pdb) files of interest into a folder (or run in stage 2B location to run all results).
2. srun or sbatch stage 2D.
3. Runs fpocket on all \_QRNAS.pdb files in the current working directory. Script will then clean up all files created and create a user-friendly summary of all pockets.
4. Once script is complete, verify that there were no errors: all pocket files generated, no errors in the .out file.

**Substage options:**

None

**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.

**Substage options:**

3AA DOCK 6 prep only

3AB DOCK 6 run only

**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.

**Substage options:**

None