Comprehensive Antibiotic Resistance Probe Design Machine (CARPDM) v1.0.1 Documentation

Input preprocessing

This step is typically not necessary when working with well-curated gene input, such as CARD database entries.

However, when working with other input that contains repeated sequences, it is helpful to align and ensure that those repeated sequences are in the same orientation. While the first step of the pipeline does remove these complementary sequences, that also means that probes will not be designed against the regions that bridge the transition of background genetic context to the repeated sequence. This may be problematic in, for example, the case of mobile elements, when that transition could be informative of placement within the genome.

Unfortunately, it is often difficult or impossible to ensure that there are no complementary sequences in the input, especially when long sequences are used as input. In this case, the pipeline can still be run. However, close attention should be paid to the output of individual target coverages and adjustments made as necessary.

Pipeline Steps

The probe design pipeline runs in several discrete steps, listed below.

Step 1: Remove complementary target sequences from input

In this step, the pipeline uses BLASTN to compare all sequences in the input fasta against each other to determine complementary regions. Since the pipeline later aggressively removes complementary probes, including these complementary regions in the design results in sparse coverage in those regions.

Instead, the program finds and removes complementary sequences by iteratively running BLASTN to compare the input sequences against themselves, finding the sequence with the highest number of complementary sequences then removing any complementary sequence >30nt long from all other input sequences. However, it leaves 10bp of the complementary sequences at either end of the excision, such that probes are still able to tile across those transitions without triggering removal during downstream complementary sequence filtering. This is repeated until no more complementary sequences remain.

This step doesn't affect probe density against the target since the complementary sequence will still be covered during probe tiling.

Step 2: Naïve Tiling

This step uses a simple script to extract every unique forward k-mer of the input sequences, where k is equal to probe length. These sequences are 'tiled,' i.e., separated by a designated length or 'step.' For example, probes may be designed with an 80nt length and 4nt step. This means that every 4nt along the template, an 80nt sequence is extracted. This is repeated

along the entire sequence. If there is a terminal sequence that goes uncovered due to the sequence length not being a multiple of the step, one more probe is generated to cover the final 80nt of the sequence, including that terminal sequence. Each probe is identified by its sha256 hash, used in sorting steps later.

Step 3: Basic Filter

In this step, perfect complements are removed from the probe set. Additionally, probes with less than a designated melting temperature are removed, as are any probes that contain ambiguous bases. Finally, if probes are meant to be used for synthesis from an oligo pool, there is a final step that removes any probes that contain an LguI cut site (5'-GAAGAGC-3' or 5'-GCTCTTC-3').

Step 4: BLAST nt Filter

BLASTN is used here to compare all probes against a local BLAST nt db. Pytaxonkit is then used to add domain names to the output. Any probe that has a number of identities >(PROBE_LENGTH* 0.625) against a viral, eukaryotic, or archaeal organism is removed so long as that number is also <(PROBE_LENGTH * 0.975). This inhibits the occurrence of off-target enrichment.

The low cutoff is based on the number of identities typically needed for a sequence to be enriched by a probe. The high cutoff is because of sequence annotation errors in the BLAST nt db. There are occasionally bacterial sequences contained within accessions annotated under other domains. Therefore, by not excluding very high-identity sequences, those sequences are not lost during this filtering step.

Step 5: Probe Complementarity Filter

Once again, BLASTN is used here. However, this time, it compares probes against themselves. If a probe has >30nt of complementarity to another probe, they are both added to consideration to be dropped from the set. After identifying candidate probes to drop, this filter iterates through every probe in order of the number of hits it has against other probes, starting with the smallest. Ties in the number of hits are broken by the sha256 hash of the probe used in the probe ID.

The complementary hits of every probe that this filter encounters are added to the set of probes to be dropped. By starting with the probes with the smallest number of hits, the program keeps a larger number of probes and therefore covers a larger sequence space. A probe's hits are only slated for removal if that probe has not yet already been added to the probes to drop.

Step 6: Probe Redundancy Filter

Here, the same BLASTN output from the complementarity filter is used; however, instead of collapsing all complementary probes, this filter instead iteratively collapses redundant probes that share a specific number of identities. For example, on the first iteration, this filter considers all probes that share 79nt of identity. It then iterates through a list of these probes, sorted by the number of hits against other probes at this identity cutoff, however, starting with the probe that has the greatest number of hits. Once again, ties are broken by the sha256 hash sequence of the

probe. It then removes any probes that share 79nt of identity with it. By starting with the probes that have the greatest number of hits, this filter removes a greater number of probes while maintaining better coverage of the input sequence space.

After removing all probes such that there are no longer any pairs that share 79nt of identity, the filter then checks if the probeset is below a user-defined probe number cutoff. This is the maximum number of probes allowed in the final design set. If not, the program reiterates with probes sharing 78nt of identity. This process repeats with decreasing levels of identity until the number of probes in the final design set is below the user-defined probe number cutoff.

Step 7 (optional): Twist oligo-pool design

This step is enabled by default and is used to create a Twist biosciences oligo-pool, such that one may synthesize their own probes, rather than ordering commercially. Typically, this results in considerable cost savings.

This step appends a premade primer to the 5′ end of the sequence, then uses primer3 to compare all possible 3′ primers and find suitable partners. Finally, the program uses BLASTN to compare all primers against the probe pool with the appended 5′ primer. The primer with the fewest number of identities to the probe pool is chosen. Finally, the program appends the reverse complement of this sequence to the 3′ end of the probes and saves the primers, as well as the oligo-pool sequences.

Step 8: Analysis

This step once again uses BLASTN to query the probes against the original design input and calculate performance statistics. By using NumPy and the positions of the query against the subject, the program adds an array of 1s and 0s, indicating matches and mismatches, respectively, to a target-length array, at the position of the alignment. This gives a simple and computationally efficient representation of the coverage of the target, form which the program calculates several statistics.

The program also plots the coverage of each target and several other useful indicators of the potential performance of the probeset

Running the Pipeline

Instructions and flags

```
usage: carpdm.py [-h] -i INPUT_FASTA -p PROBE_NUM_CUTOFF -s TILING_STEP
                 [-l PROBE_LENGTH] [-m MELT_TEMP] [-b BASENAME] [-d OUTPUT_DIR]
                 [-t NUM_THREADS] [-n] [-v]

Design probes against an input fasta according to design parameters, then pass through
several filters to remove off-target enrichment (nt filter) and probe redundancy (self
filter)

options:
  -h, --help            show this help message and exit
  -i INPUT_FASTA, --input_fasta INPUT_FASTA
```

```
                        Path to the fasta that contains sequences against which
                        probes are to be designed. Required
  -p PROBE_NUM_CUTOFF, --probe_num_cutoff PROBE_NUM_CUTOFF
                        Maximum number of final probes allowed in the probeset. The
                        redundancy filter will iterate until it reaches below this
                        cutoff. Required
  -s TILING_STEP, --tiling_step TILING_STEP
                        Number of bases by which to offset initial probe tiling.
                        Higher values make computation less intense, though may
                        result in sparser coverage. Required
  -l PROBE_LENGTH, --probe_length PROBE_LENGTH
                        Probe length to create. Note that if >80 (default), probes
                        will be too long for the base price of a Twist oligo pool
                        after addition of amplification and transcription primers for
                        in-house synthesis. Default = 80
  -m MELT_TEMP, --melt_temp MELT_TEMP
                        Minimum melting temperature for probes to be considered
                        during basic filter. Default = 50
  -b BASENAME, --basename BASENAME
                        File basename
  -d OUTPUT_DIR, --output_dir OUTPUT_DIR
                        Output directory
  -t NUM_THREADS, --num_threads NUM_THREADS
                        Number of threads to be used during BLAST searching
  -n, --no_o_pool       If included, omits steps to design template oligo pool. Only
                        include if ordering RNA probes directly from the manufacturer.
  -v, --version         show program's version number and exit
```

Examples

Run the pipeline with the following conditions (allCARD conditions):

- 4nt step
- 40000 probe number cutoff
- 40 threads
- Default probe length (80) and melt temp (50)

```
carpdm.py -i /path/to/card_nucleotide_protein_homolog_model.fasta -b allcard -d
allCARD_probe_design -t 40
```

clinicalCARD was constructed using the same conditions; however, a different, input file.