

Probe Design Using Exact Repeat Count



August 8th, 2007

Aaron Arvey



Outline

- Qualities of good probes
- Past Work
- Our Algorithm



Oligo Probes

- Sequences are controlled
 - In contrast to cDNA/other in vitro methods
- More genomes are getting sequenced, enabling oligo probe usage
- Large genomes are hard
 - Lack of compactness leads to repeats
 - Probes may bind to similar sequences
 - Specificity vs sensitivity



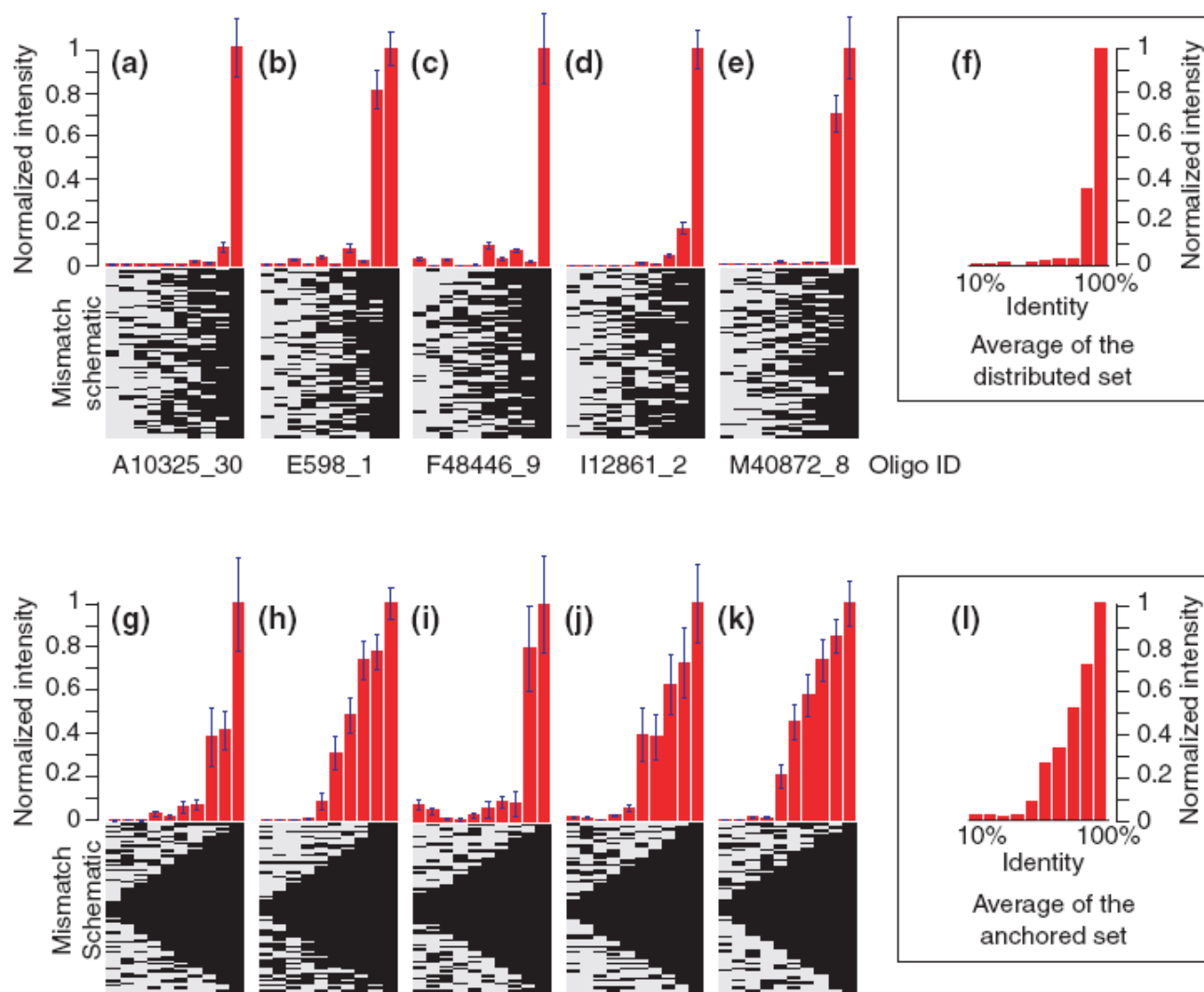
Good Probes

- “Completely” unique
 - No nonspecific binding to similar sequences (w.r.t. binding energy)
 - Regions (e.g. endpoints) are more important
- No G-quartets
- Ends in C/G (less “breathing”)
- No hairpin structures
- Fairly balanced AT% vs GC%

Empirically Specific Probes

(For Microarrays)

- “50-mer probe with 15-base, 20-base, or 35-base stretch with nontargets, has approximately 1%, 4%, or 50% of the target signal intensity (Kane et al, Hughes et al)”
- “...at probe-target identity of 75% or greater, minimal free energy is closely correlated to probe-target identity (data not shown).” (He et al, 2005)



Each graph shows intensity of microarray.
 Right column shows perfect match.
 Left columns show mismatches (in gray).
 Taken from Bozdech et al 2003 (De Risi Lab)



Do Good Probes Exist?

- A completely unique sequence is likely to be fairly uniformly random
 - From this we can say that a unique sequence will likely
 - lack internal repeats
 - have low probability of forming hairpin structure
 - have fairly balanced AC% vs. GC%
- $\text{Prob}(\text{G quartet}) = .25^4 = 1/256$
- Good sequences are likely to exist



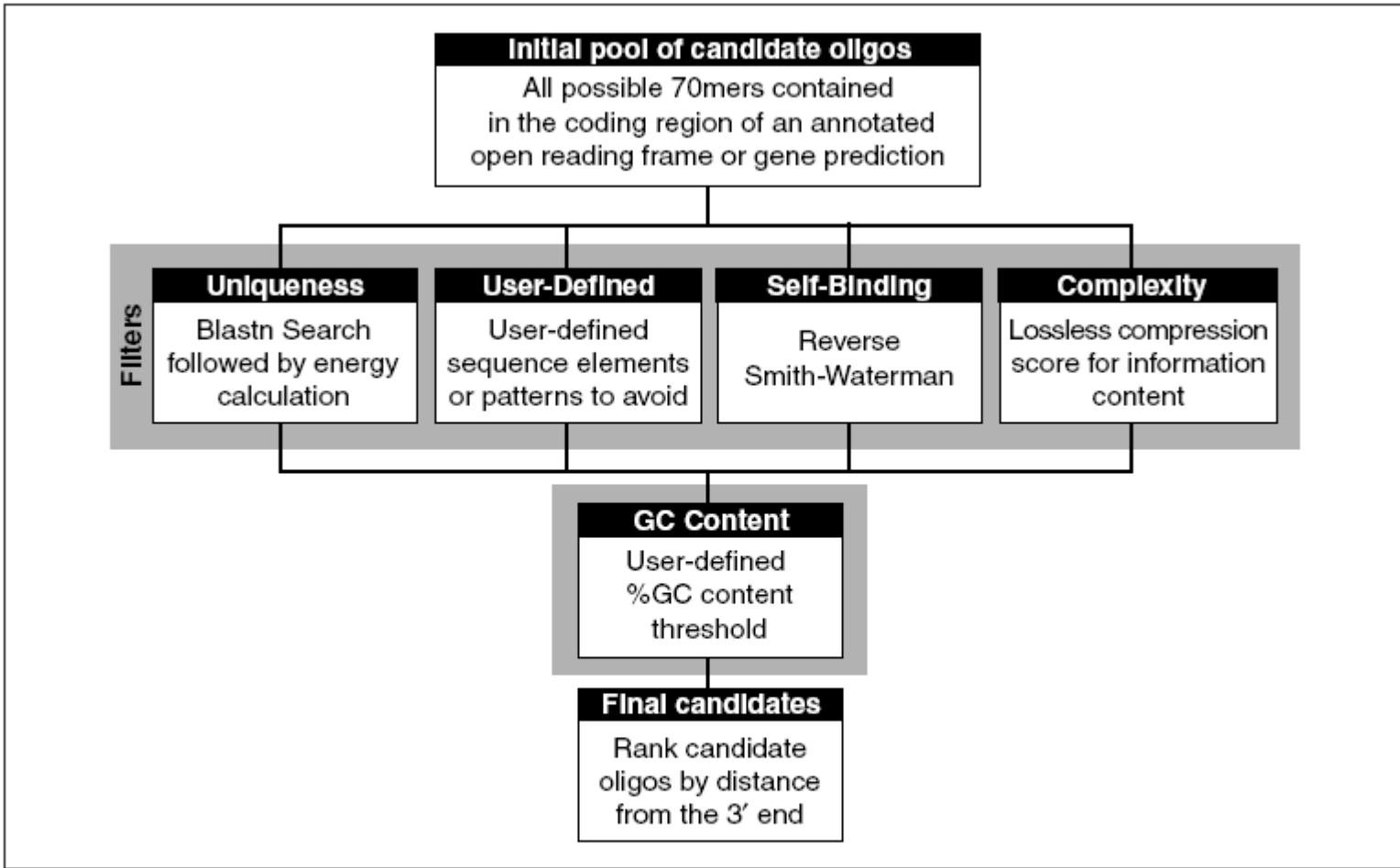
Finding Good Probes

- Computationally naïve approach
 - BLAST every possible kmer probe, given a desired target sequence
 - May take days to complete single probe
- Heuristics
 - Use kmer substring uniqueness.
 - D. mel: 12mer, 15mer
H. sap: 16mer, 18mer, 20mer, 24mer

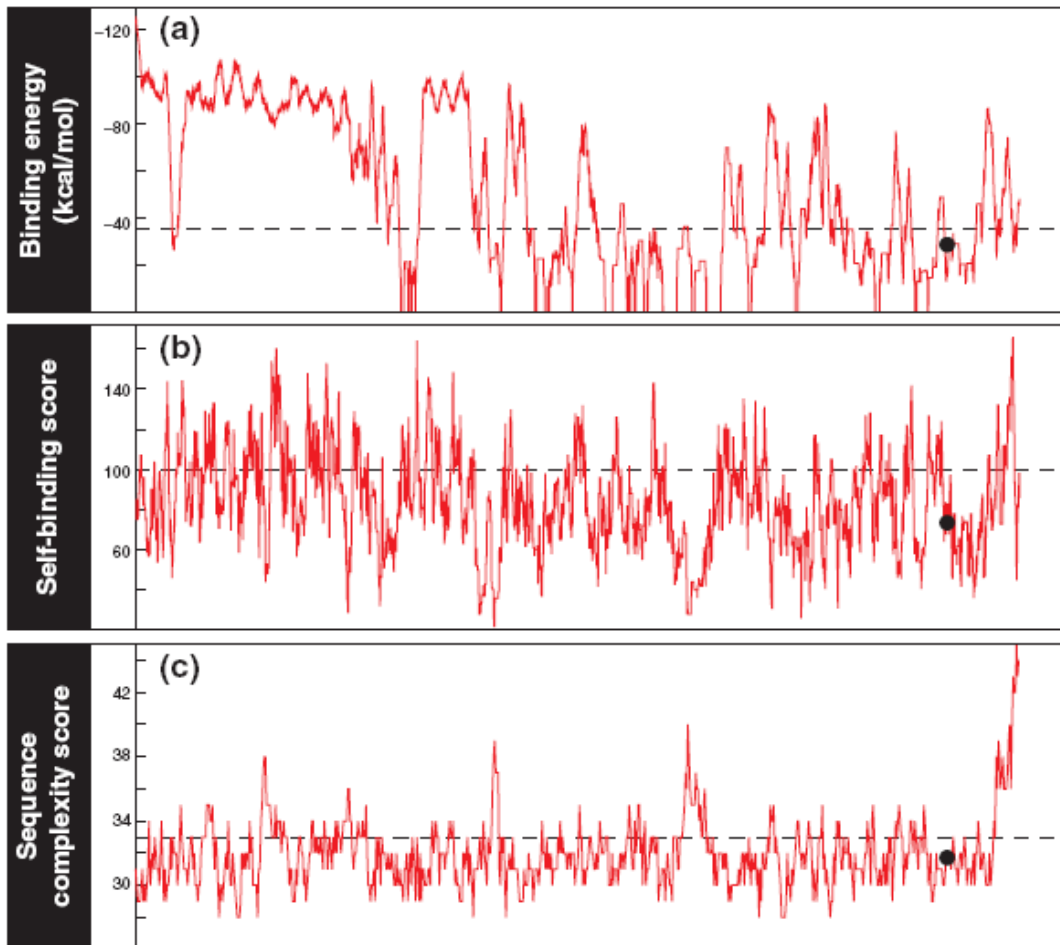
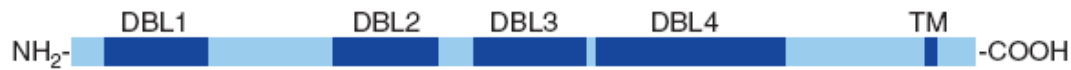


Past Algorithmic Approaches

- BLAST searches (DeRisi '03, others)
- Suffix array/tree (Li & Stormo 2000)
- Burrows-Wheeler transform (CSHL 2005)
 - Sorting and compressing of suffix array
- Almost entirely for microarray studies
 - The problem in the *in situ* domain is harder



Selection of probe given a sequence
 Taken from Bozdech et al 2003 (De Risi Lab)



← Binding Energy of nearest BLAST search

← Self-Binding (Hairpin) Score

← "Sequence Complexity" (Compressibility)

Taken from Bozdech et al 2003 (De Risi Lab)



Times for past algorithms

- Suffix Array (Li & Stormo 2001)
 - 4 days for 4-mismatch distance of 24mer probes on *E. coli* (12MB)
 - Would be months for 50mers on *E. coli*
 - Would be centuries for 50mers on humans
- BLAST searches (DeRisi Lab 2003)
 - 12 hours for 70mer probes on *P. falciparum* 12MB
 - Would be weeks/months for humans



Times for past algorithms

- Burrows-Wheeler Transform (CSHL 2005)
 - Requires 6-7 days to construct dictionary
 - Able to store on disk and use disk seeks
 - After preprocessing, queries are as fast as ours
- Suffix Trees
 - Requires 40-50GB RAM
 - Likely to be slightly slower than our algorithm



Our Idea

- Use a heuristic to find good regions
 - Find subsequence uniqueness for 15-20mers
 - Use Bloom Filter (probabilistic)
 - Use boolean “is repeated”
 - Use exact repeat counts
- Use BLAST to find second most homologous sequence and check binding energy to probe



What We Want to Know

AGCTAGCTAGTCAAAGGT	(UNIQUE)
GCTAGCTAGTCAAAGGTA	(UNIQUE)
CTAGCTAGTCAAAGGTAA	(UNIQUE)
TAGCTAGTCAAAGGTAAT	(REPEATED)
AGCTAGTCAAAGGTAATA	(UNIQUE)
GCTAGTCAAAGGTAATAG	(UNIQUE)
CTAGTCAAAGGTAATAGG	(UNIQUE)

- **S=TAG...AAT is repeated**
- **AS, GS, or TS exist**
- **ST, SG, or SC exist**

Design

Preprocessing

Print all
kmers
to file

Sample kmer
cumulative
distribution

Create γ^d
files for
CDF sorting

Count records
in sorted files
sequentially

Kmer
Input

Sequence
Input

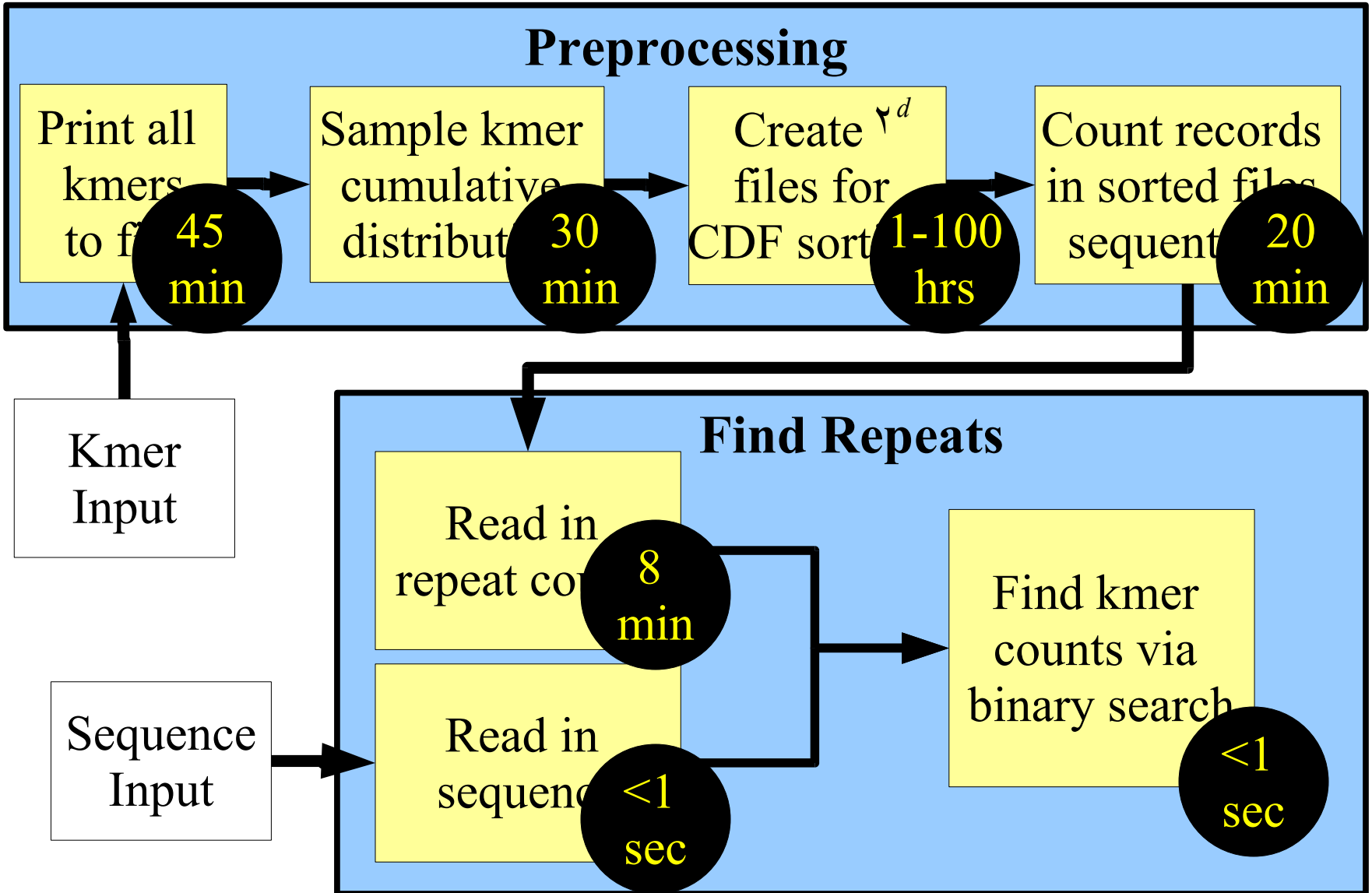
Read in
repeat counts

Read in
sequence

Find Repeats

Find kmer
counts via
binary search

Time for *H. sapiens*



Size for *H. sapiens*

Preprocessing

Print all kmers to file

3GB
Disk
1 File

Sample kmer cumulative distribution

20GB
Disk
1 File

Create γ^d files for CDF

20GB
Disk
500 Files

Count records in sorted files sequentially

1/2-2GB
Disk
1 File

Kmer Input

Sequence Input

Find Repeats

Read in repeat counts

1/2-2GB
RAM

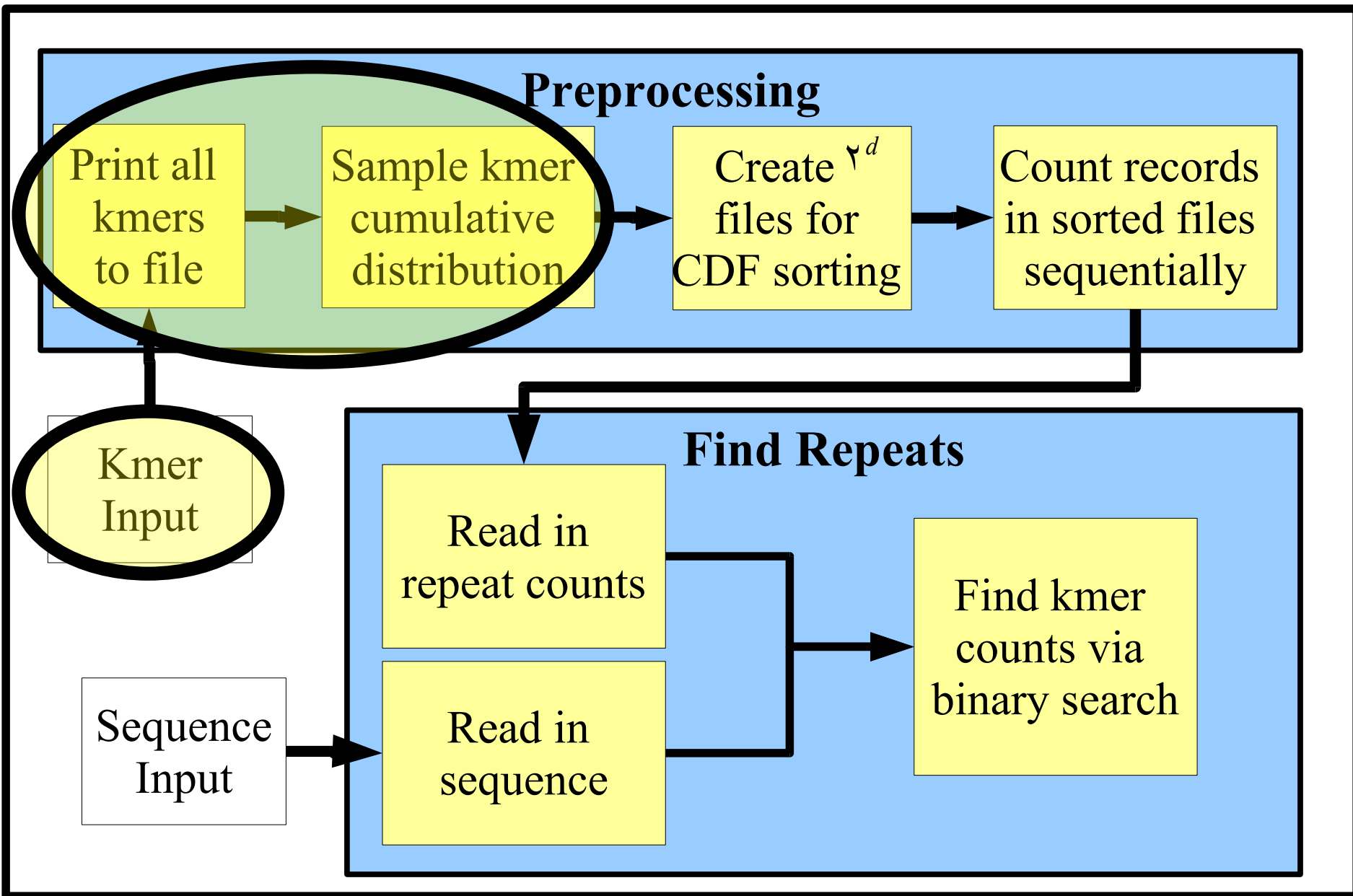
Read in sequence

10KB
RAM

Find kmer counts via binary search

400KB
RAM

Algorithm





Preprocessing – Printing Kmers

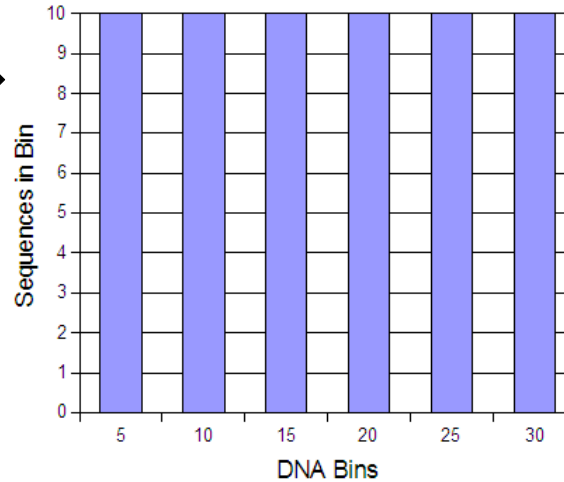
- A genome file is read in (3GB)
- Every kmer (including overlaps) is printed to an output file (10-20GB)
- Kmers are sampled to create an empirical cumulative distribution (CDF) and put into files
 - Goal is to uniformly sample as to avoid abnormally large files

Why Sampling of CDF?

What we'd like



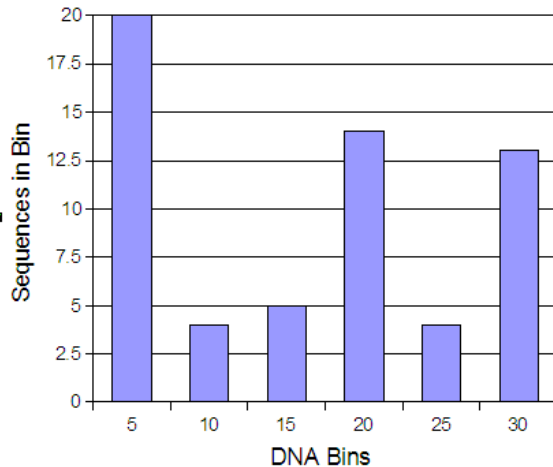
Uniform Distribution



What we get



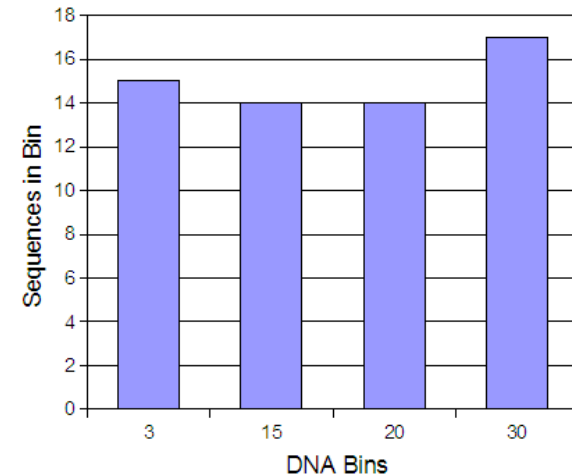
Actual Distribution



What we make



Sampled Distribution



Algorithm

Preprocessing

Print all
kmers
to file

Sample kmer
cumulative
distribution

Create γ^d
files for
CDF sorting

Count records
in sorted files
sequentially

Kmer
Input

Sequence
Input

Read in
repeat counts

Read in
sequence

Find Repeats

Find kmer
counts via
binary search



Preprocessing – Sorting Kmers

- Each chunk of the CDF is sorted
- All repeated kmers are adjacent
- Read the sorted files sequentially
- Output repeated kmers and number of times repeated to output file

Algorithm

Preprocessing

Print all
kmers
to file

Sample kmer
cumulative
distribution

Create γ^d
files for
CDF sorting

Count records
in sorted files
sequentially

Kmer
Input

Sequence
Input

Read in
repeat counts

Read in
sequence

Find Repeats

Find kmer
counts via
binary search



Finding Repeats – Specifications

- Input a sequence (e.g. large intron)
 - Specify plus/minus strand
 - Specify granularity of kmer
- Read in repeat file from preprocessing
 - Read the repeat
 - Read the number of times it was repeated

Algorithm

Preprocessing

Print all
kmers
to file

Sample kmer
cumulative
distribution

Create γ^d
files for
CDF sorting

Count records
in sorted files
sequentially

Kmer
Input

Sequence
Input

Read in
repeat counts

Read in
sequence

Find Repeats

Find kmer
counts via
binary search



Finding Repeats – Searching

- For each of the kmers in the sequence, see if it is in the repeat file
 - Since the repeat file is sorted, we can do a binary search
 - Each binary search takes <0.00001 seconds
- Output a chart showing the repeat structure of the sequence



Questions?
