wellcome
connecting
science

FastK:

A K-mer Counter & Read Profiler

and its

Applications

Gene Myers

Sanger K-mer Class
June 3, 2025

What It Can Do:

wellcome connecting science

A histogram of k-mer frequencies (always)

A sorted table of k-mers & their counts (-t)

For each read/contig/sequence, a k-mer profile (-p)

FastK [-k<int(40)>] [-t<int(1)]> [-p] …

e.g. FastK -k21 -t foo.fasta

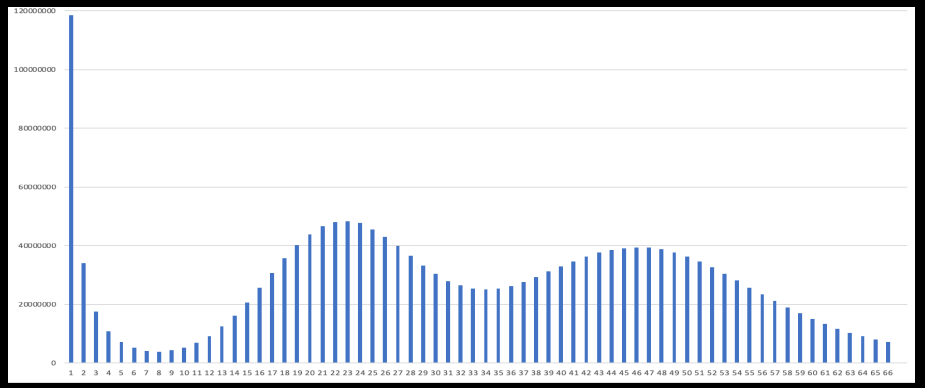
produce 21-mer count histogram and table of all 21-mers

e.g. FastK -t4 -p foo.bam

produce 40-mer count histogram, table of all 40-mers occuring ≧ 4 times, and a profile of every sequence

FastK Outputs

- foo.hist: array of # of kmers occuring f times, for f ∈ [1..32,767]



```
    . . .
59023: agaatccggaagaatacccgggaaaagatagaatccgtgcg =     5
59024: agaatccggaagactaccgggaaaagatagaatccgtgcg =    23
59025: agaatccggaagagtaccgggaaaaggaagaatccgtgcg =    18
59026: agaatccggacgactacccgtaaatggtagaatccgtgcg =     4
59027: agaatccggacgagtaccgggaaaaggtagaatccgtacg =     6
59028: agaatccggacgagtatcaggaaaaggtagaatccgtacg =    15
59029: agaatccggacgattaccgggaaaaggtagaacacgtacg =     5
59030: agaatccggatgactaccgggaaaagatagaatccgtgcg =    32
59031: agaatccggatgactaccgggaaaagatagaatccgtgct =     5
59032: agaatccggatgactaccgggaaagatagaatccgtgcga =     4
59033: agaatccggatgactacctggaaaagatagaatccgtgcc =     5
59034: agaatccggatgactacctggaaaagatagaatccgtgcg =     9
59035: agaatccggatgactacctggaaaagatagaattcgtgcg =     4
59036: agaatccgggagaataccgaaaagtcgtaaaaatcagccg =     5
59037: agaatccgggcgaatacccgtaaatggtagaatccgtgcg =    12
59038: agaatccgggcgaatacgaaaaaacgtaaaattccgccg =    10
59039: agaatccgggcgaatacgaaaaatcgtcaaattccgccg =     4
59040: agaatccgggcgaatacgaaaaatcttaaaattccgccc =    16
59041: agaatccgggcgactacccgtaaatggtagaatccgtgcg =   134
59042: agaatccgggcgactacccgtaaatggtagaatccttgcg =    14
59043: agaatccgggcgactaccgggaaaaggtagaatttatgct =     4
59044: agaatccgggcgactaccgggtaaaggtagaatccgtgcg =    86
59045: agaatccgggcgactaccgggtaaaggtagactccgtgcg =    37
59046: agaatccgggcgactaccgggtaaagttagaatccgtgcg =     5
59047: agaatccgggcgagtaactagaaaaggtagaatccatgcc =     4
59048: agaatccgggcgagtaccggaaaaaggtagaatccatgcg =    29
59049: agaatccgggcgagtaccgggaaaaggtagaatccatgcg =    72
59050: agaatccgggcgagtaccgggaaaaggtagaatccgggcg =     5
59051: agaatccgggcgagtaccgggaaaaggtagaatccgtacg =     9
59052: agaatccgggcgagtaccgggaaaaggtagaatccgtgcg =     7
    . . .
```

- foo.ktab: lexicographically sorted array of all unique k-mers and their counts
- foo.prof: for each sequence in order, the count of each k-mer along the sequence.

wellcome connecting science

Where is it?

https://github.com/thegenemyers/FASTK

Build: Type “make” (HTSLIB & LIBDEFLATE included)

Documentation: README.md @ github

FastK: A K-mer counter (for HQ assembly data sets)

wellcome connecting science

Author: Gene Myers
First: July 22, 2020
Current: April 18, 2021

GitHub @ thegenemyers/FASTK

- Command Line
  - FastK
  - Fastrm, Fastcp, & Fastmv ← Tables & Profiles use hidden files
  - Fastmerge
  - Fastcat ← Facilitate CPU parallelism
- HPC Operation
- Core Applications
  - Histex: Display a FastK histogram or convert to 1-code
  - Tabex: List, Check, find a k-mer in a FastK table, or convert to 1-code
  - Profex: Display a FastK profile or convert to 1-code
  - Logex: Combine kmer,count tables with logical expressions & filter with count cutoffs ← General Table Logic
  - Symmex: Produce a symmetric k-mer table from a canonical one
  - KmerMap: Produce a .bed file showing all the regions in a target covered by a set of k-mers

↕ Basic examples of how to access the 3 output objects

- C-Library Interface
  - K-mer Histogram Class
  - K-mer Table Class
  - K-mer Stream Class ← Streams through a table using fixed memory buffers. Many op's can be so accomplished given tables are sorted.
  - K-mer Profile Class ← Profiles are loaded & uncompressed on demand.
- File Encodings
  - `.hist` : K-mer Histogram File
  - `.ktab` : K-mer Table Files
  - `.prof` : K-mer Profile Files

Input

wellcome connecting science

Accepts series of sam, bam, cram, fasta, or fastq (optionally gzip'd) as input but all must be the same type.

<source>[.cram|.{bs}am|.f[ast]{aq}[.gz]] …

For the source you do not need to specify the extension, FastK will figure it out, e.g. FastK ../blue/foo will run on foo.bam if that is what is in the directory ../blue.

Order of search is cram > bam > sam > fa > fq > fa.gz > fq.gz

Only required argument(s). Optional arguments (those begining with a '-') may be in any order:

e.g. FastK -k21 foo1 -t3 foo2 -T12

Gzip'd files cannot be partitioned across threads, at best 1 .gz file per thread. Gzip is ancient, you should really use bam or cram.

Output

wellcome connecting science

Normally, the name of any .hist, .ktab, and .prof files has the same root path as the source:

e.g. FastK foo -t -p produces foo.hist, foo.ktab, and foo.prof

But can specify an output path explicity with the -N option

… [-N<path>] …

e.g. FastK foo -t3 -N../blue/boo produces boo.hist and boo.ktab
in the directory ../blue

Implicit outputs are also placed in the same directory as the source:

e.g. FastK ../blue/foo produces foo.hist in directory ../blue

What It Can Do:

wellcome connecting science

Can count in hoco space (-c)

Will ignore bar-code prefixes (-bc)

Relative profiles: k-mer counts from a distinct source mapped onto input reads (e.g. Illumina onto contigs)

… [-c] [-bc<int>] [-p:<FastK table>] …

What It Can Do:

wellcome connecting science

Can count in hoco space (-c)

In homopolymer compression all homopolymers are reduced to a single base: e.g. gtaaaattgccctaatgg = gtatgctatg

Most sequencing errors are the length of homopolymer runs, e.g. for HiFi reads 80% or so.

Running in hoco space implies fewer error k-mers and reduces the data set by 1/3rd.

Will ignore bar-code prefixes (-bc#)

Many data sets contain bar-code sequences at the start of a read, e.g. multiplexed HiFi runs.

-bc<n> asks FastK to ignore the first n bases of each read/sequence.

What It Can Do:

wellcome connecting science

Relative profiles: k-mer counts from a distinct source mapped onto input reads (e.g. Illumina onto contigs)

FastK -p:Illumina.ktab Assembly.fasta -NAsm.Ill

Profiles for scaffolds in Assembly.fasta are for the counts of the k-mers in FastK table Illumina.ktab.

FastK -p:Mother.ktab Child.fasta -NMat
FastK -p:Father.ktab Child.fasta -NPat

Counts of mother & father Illumina projected onto Pacbio reads of their child

Counts can be 0!

Resources/Limits

wellcome connecting science

Threaded up to 256 (-T, default 4)

Has a modest fixed memory ceiling (-M, default 12GB)

Large scratch disk used in specifiable directory (-P, default /tmp)

… [-M<int(12)>] [-T<int(4)>] [-P<dir(/tmp)>] …

Currently -M must be ≦ 128GB, and -k ≦ 128

Works best with 4-16 threads, and memory ~16GB, more doesn't improve speed much

When using a cluster it is very important that -P refer to the local disk of the cluster node.

Hidden Files (achtung!)

wellcome connecting science

.ktab & .prof are big and involve hidden files:

X.ktab & X.prof =“stub” files, the data is in:

.X.ktab.1, .X.ktab.2, … .X.ktab.# = threads used
.X.pidx.1, .X.pidx.2, … .X.pidx.#
.X.prof.1, .X.prof.2, … .X.prof.#

Fast(rm|mv|cp) to ease manipulation of said.

e.g. Fastmv X ../blue/Y moves any FastK file with root X to directory ../blue and renames it Y.

Semantics and options are the same as UNIX rm, mv, and cp.

Invalid K-mers

N's occur frequently (gaps in assemblies, poor call in Illumina).

In general, a k-mer is invalid if it contains a letter other than a,c,g,t.

Invalid k-mers do not appear in tables and are not counted.
Invalid k-mers have count 0 in profiles.

Truncated Counts

FastK only counts to $2^{15}-1$ = 32,767.

But it does determine & record in the .hist how many k-mers have this or higher count, and the total # of instances of these k-mers.

So it knows how many k-mers and k-mer instances are in the input data.

K-mers with count ≧ 32,767 are in the .ktab, with count 32,767.

wellcome
connecting
science

EXERCISES:

Get it, Make it, Run it

wellcome
connecting
science

FastK:

Some Algorithmic Ideas

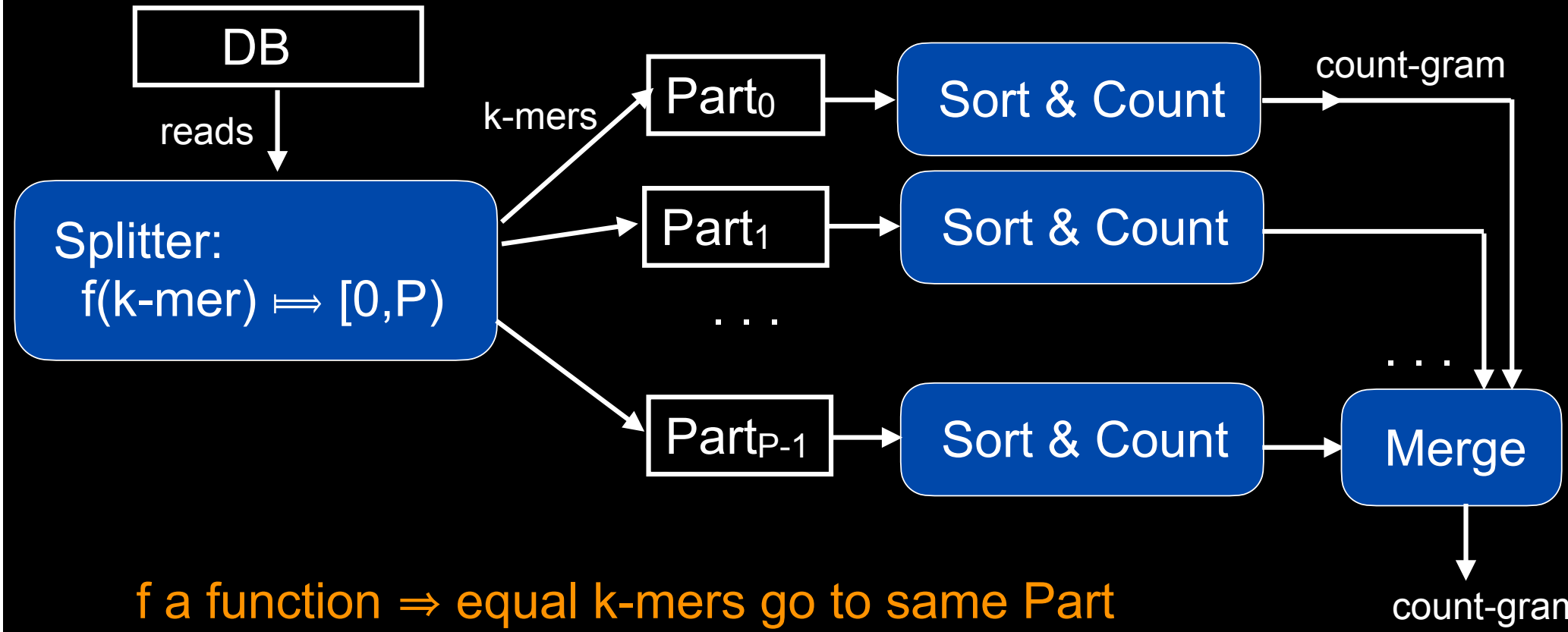
For your amusement

Disk-Based Counting Schema (e.g. KMC3)

wellcome connecting science

Have DB of Nbp and R reads ⇒ N-(k-1)R k-mers

Want each sort to take ≤ M bytes of memory, say 12GB



f a function ⇒ equal k-mers go to same Part

Pick P so that each sort fits in say 12GB of memory

Use an MSD sort !!

Minimizers (Review)

wellcome connecting science

The canonical value of w is:

$C(w) = \min(w, w^c)$ (as a base 4 #)

The m-minimizer of w is:

$M_m(w)$ = m-mer of w with smallest C-value (for m < |w|)

Given k and m<k, w is a (k,m) super-mer iff:

all k-mers of w share the same m-minimizer instance

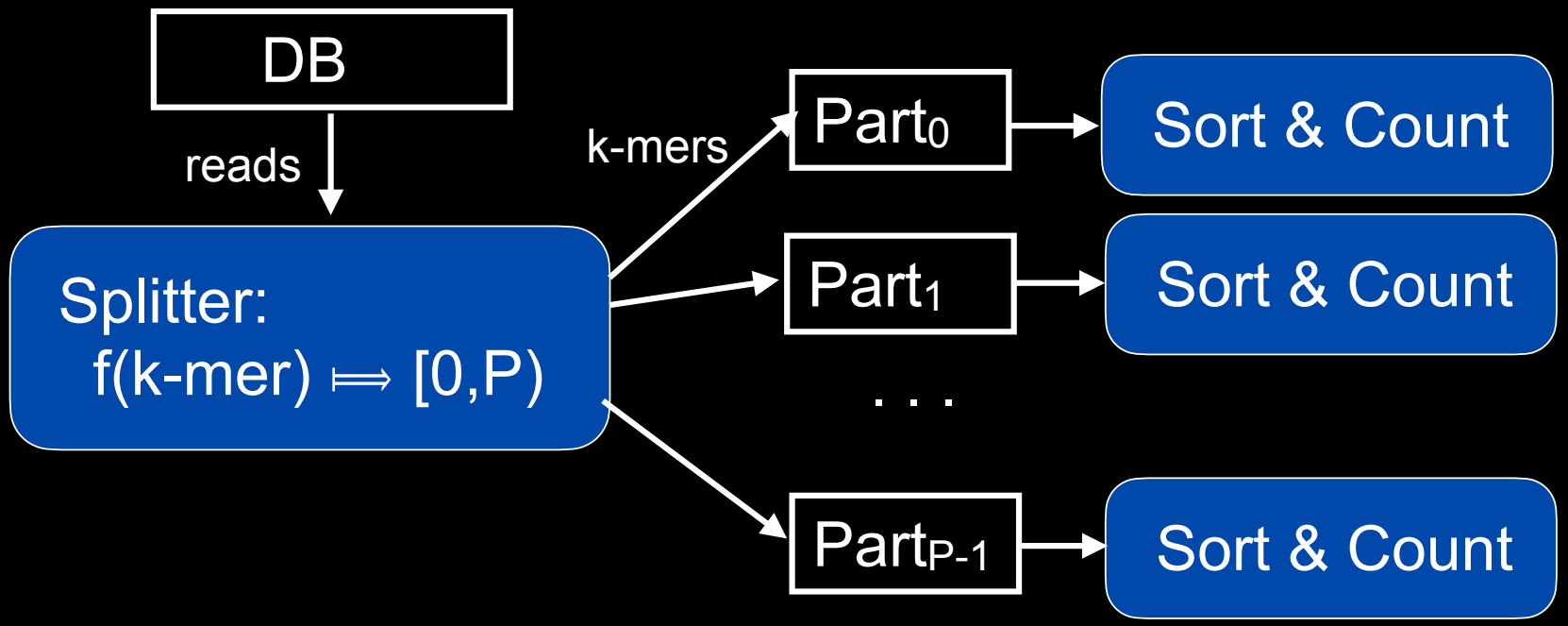
e.g. `agtgcatgcttacgtgcaa` (11,3)

The largest possible super-mer is of length 2k-m.

Consecutive k-mers tend to share the same m-minimzer.

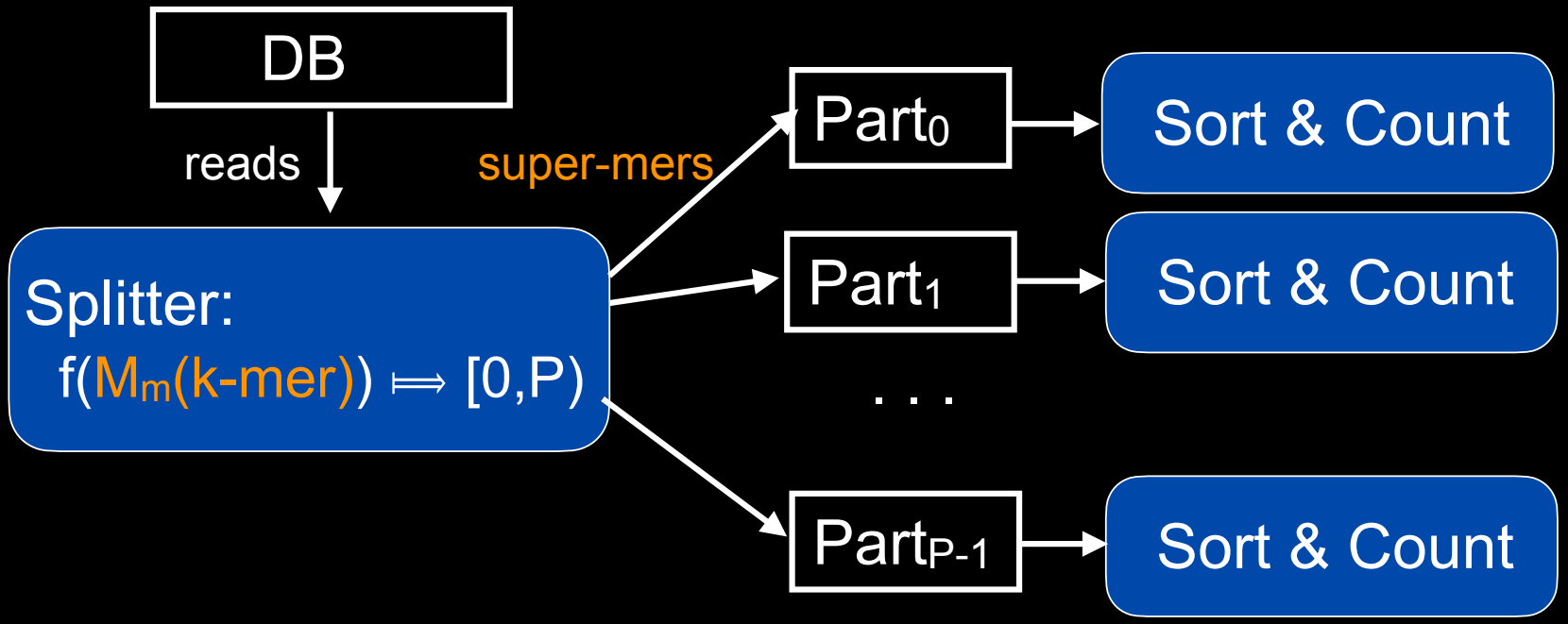
Supermers average is about (k-m)/2 k-mers.

Disk-Based Counting Schema



Disk-Based Counting Schema

wellcome connecting science



Every k-mer in a super-mer has the same minimizer, so distribute super-mers not k-mers.

Super-mers are more compact then k-mers

e.g. k,m = 40,7 then typical super-mer has 16 k-mers

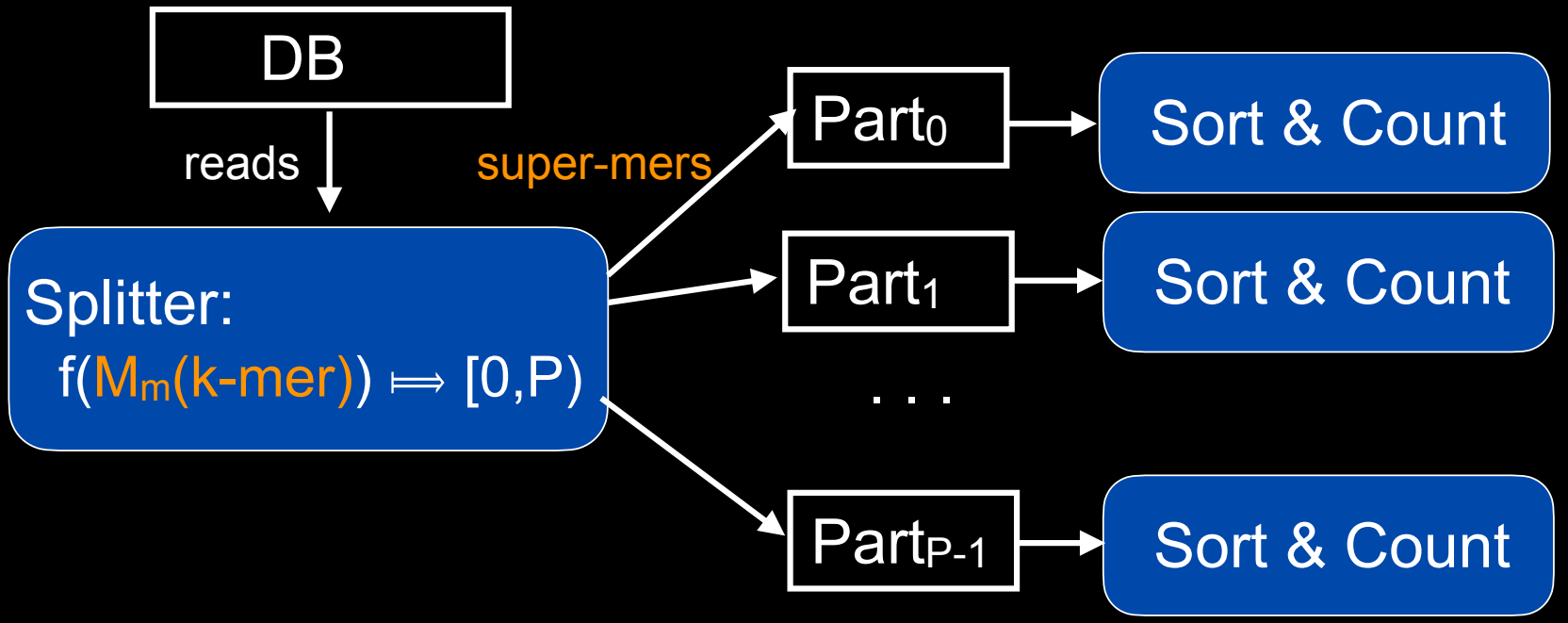
e.g., k+15 bp
vs.
16k bp

Indeed, send k+(k-m)/2 data per (k-m)/2 k-mers

⇒ ~3bp per k-mer ⇒ O(N) not O(Nk) !

Disk-Based Counting Schema

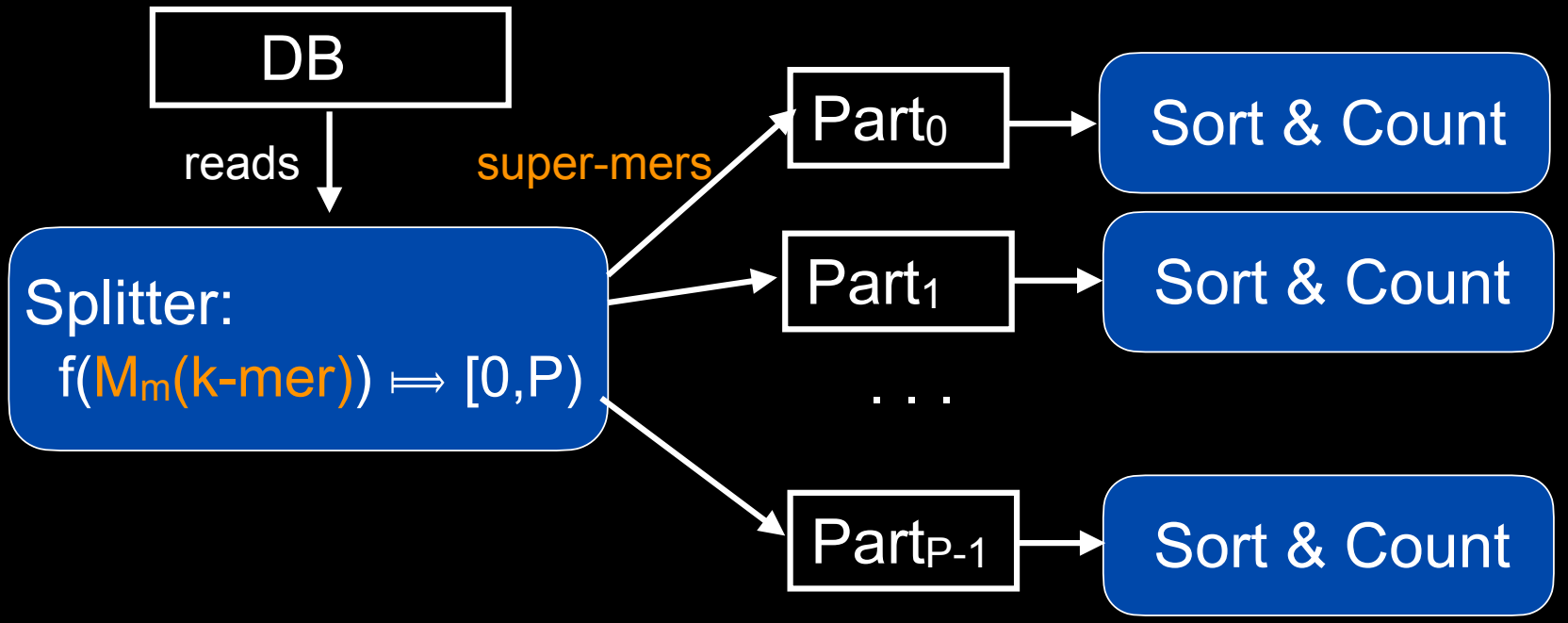
wellcome connecting science



Still need f: There are $O(4^m)$ >> P canonical m-mers

Disk-Based Counting Schema

wellcome connecting science

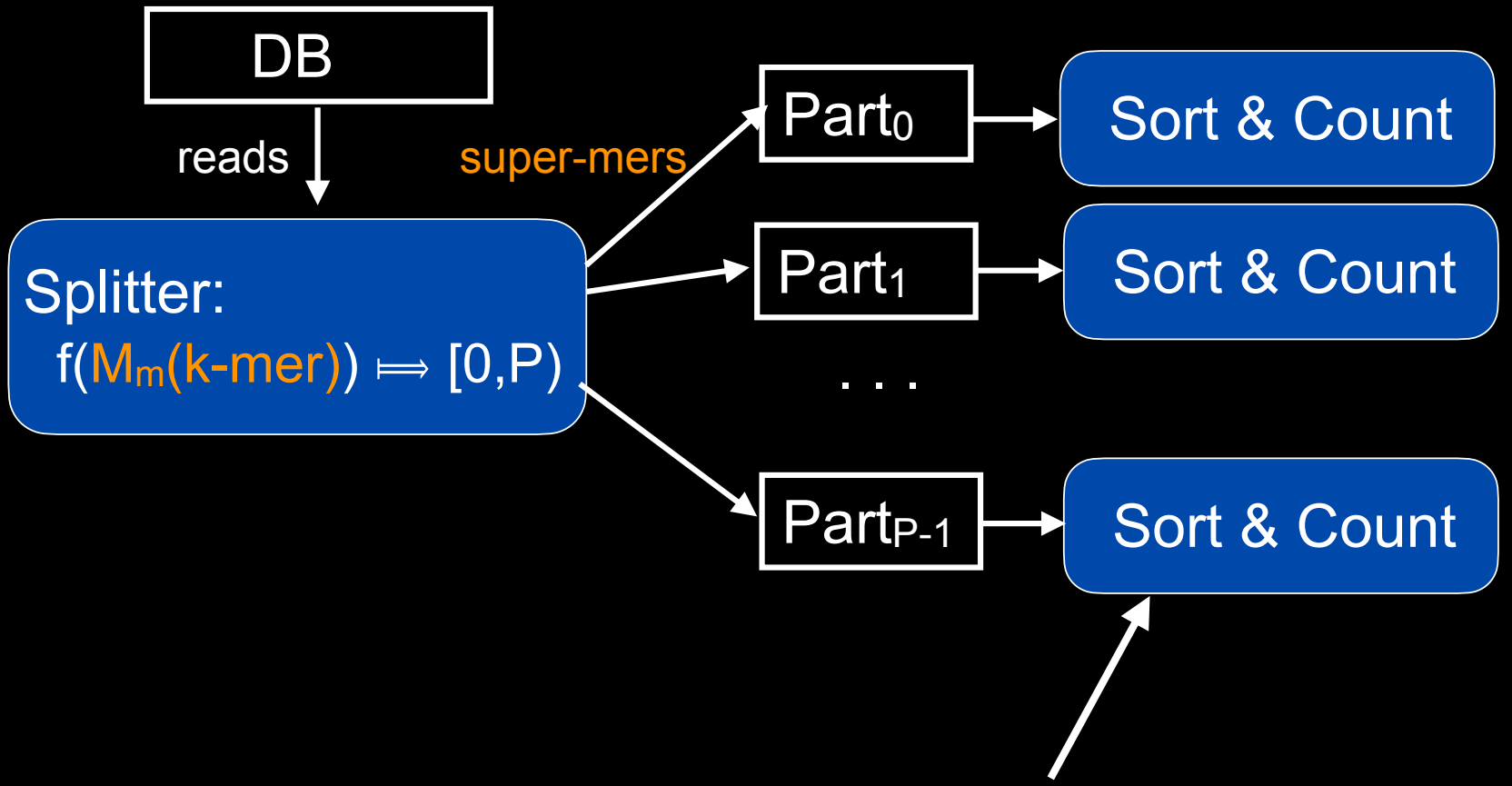


Normally, unpack super-mers into k-mers & sort

For a low-error rate data set, 1% or less, super-mers often contain no errors, implying they occur many times depending on coverage of the underlying genome.

Disk-Based Counting Schema

wellcome connecting science



Sort super-mers, determining count for each.

Build list of weighted k-mers from each distinct super-mer and sort.

Accumulate weights of equal k-mers.

Cabernet Example, k,m = 40,7: 0.2% error

wellcome connecting science

```
Phase 2: Sorting & Counting K-mers in each Super-mer File

  Processing CBS.0.T0-3 representing 999,317,304 40-mers:
    Sorting 50,978,047 super-mers (19.6X reduction)
    Sorting 119,114,082 weighted k-mers (8.4X reduction))

  Processing CBS.1.T0-3 representing 992,136,060 40-mers:
    Sorting 63,594,675 super-mers (15.6X reduction)
    Sorting 110,800,071 weighted k-mers (9.0X reduction))

  Processing CBS.2.T0-3 representing 998,204,702 40-mers:
    Sorting 50,239,140 super-mers (19.9X reduction)
    Sorting 128,874,573 weighted k-mers (7.7X reduction))

  Processing CBS.3.T0-3 representing 997,095,912 40-mers:
    Sorting 87,160,157 super-mers (11.4X reduction)
    Sorting 110,628,297 weighted k-mers (9.0X reduction))

   .   .   .
```

⇒ about 4.5X over k-mer sort

Cichlid Example, k,m = 20,7: 10% error

wellcome connecting science

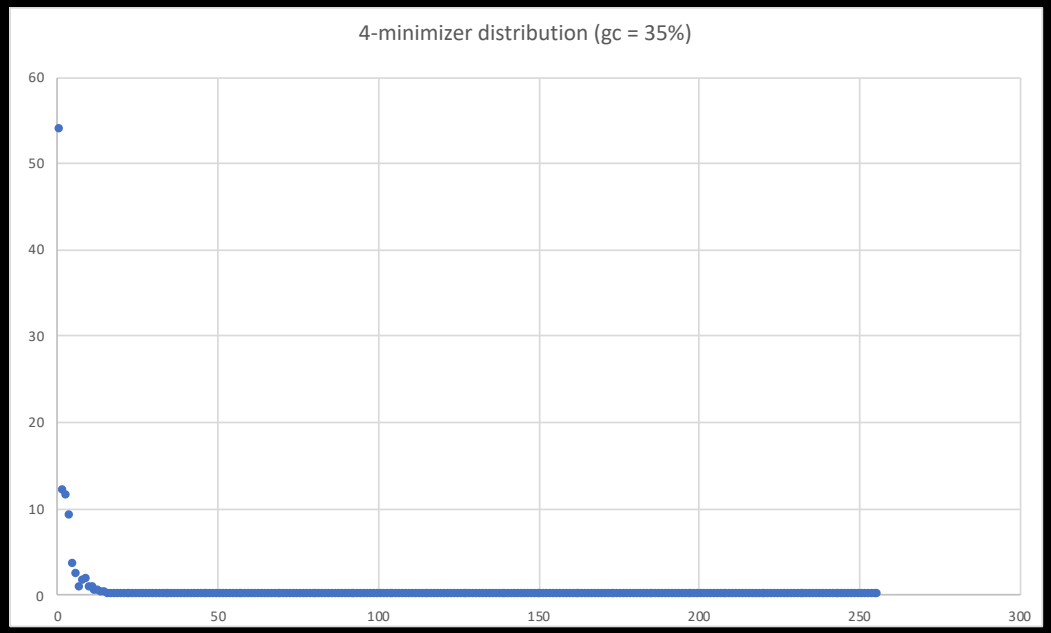
```
Phase 2: Sorting & Counting K-mers in each Super-mer File

  Processing CICHLID.0.T0-3 representing 999,317,304 40-mers:
    Sorting   228,306,350 super-mers (7.3X reduction)
    Sorting 1,566,808,827 weighted k-mers (1.1X reduction))

  Processing CICHLID.1.T0-3 representing 992,136,060 40-mers:
    Sorting   235,557,318 super-mers (6.9X reduction)
    Sorting 1,523,110,422 weighted k-mers (1.1X reduction))

   .   .   .
```

Super-mer sort is wasted time as only 1.1X reduction in k-mers to sort.

KMC3's k-mer sorter is about 1.5-7X faster using an idea I did not incorporate into FastK so it is about 2X on such data. But …

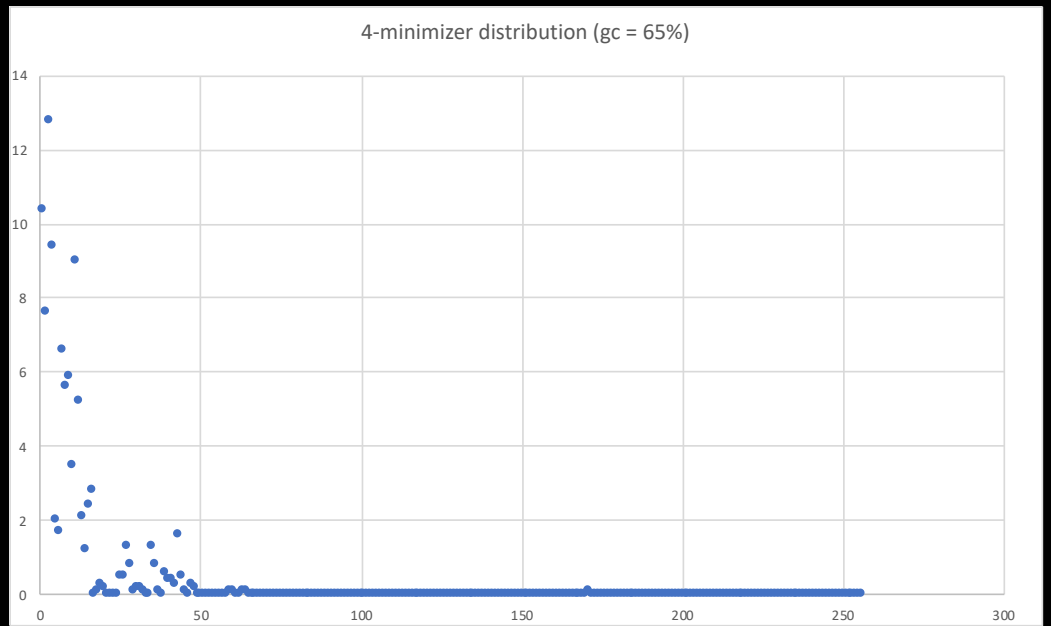
Minimizer Distributions are (very) Skewed.

Consider k = 40, m = 4, Cabernet Sauvignon => gc% = 35,



0 = 'aaaa' is the minimizer for
54% of 40-mers!

a -> 0 (0.318858)
c -> 1 (0.180059)
g -> 2 (0.179884)
t -> 3 (0.321199)

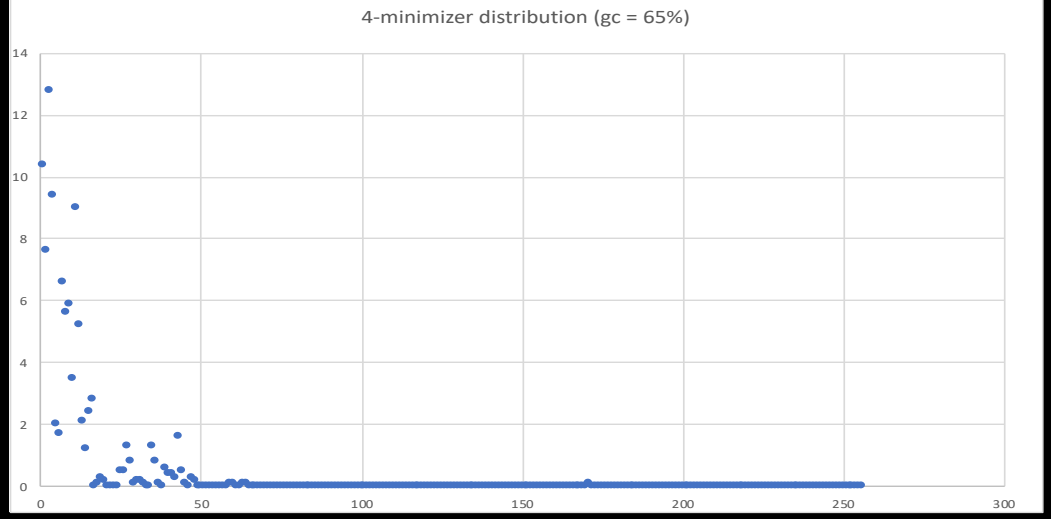


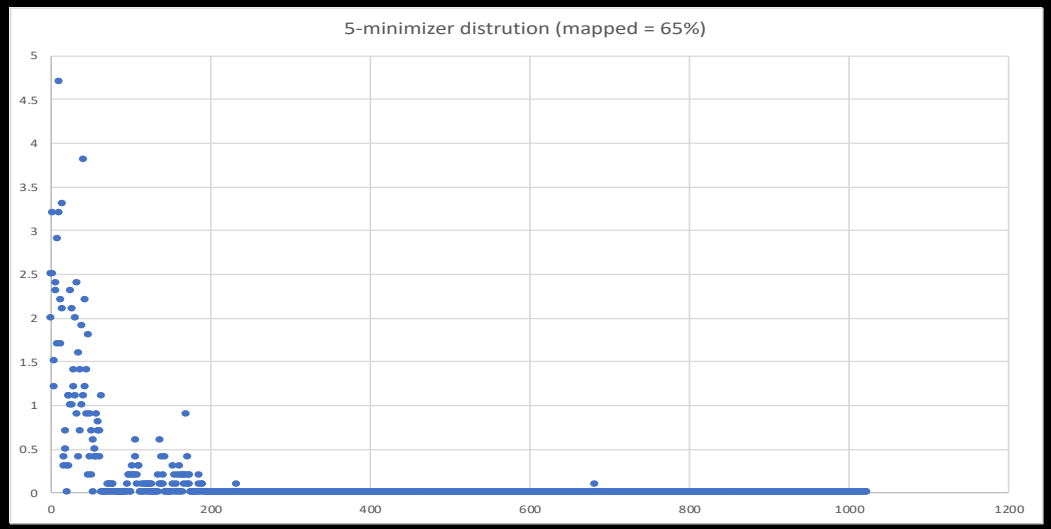
Recode/map bases in order of frequency

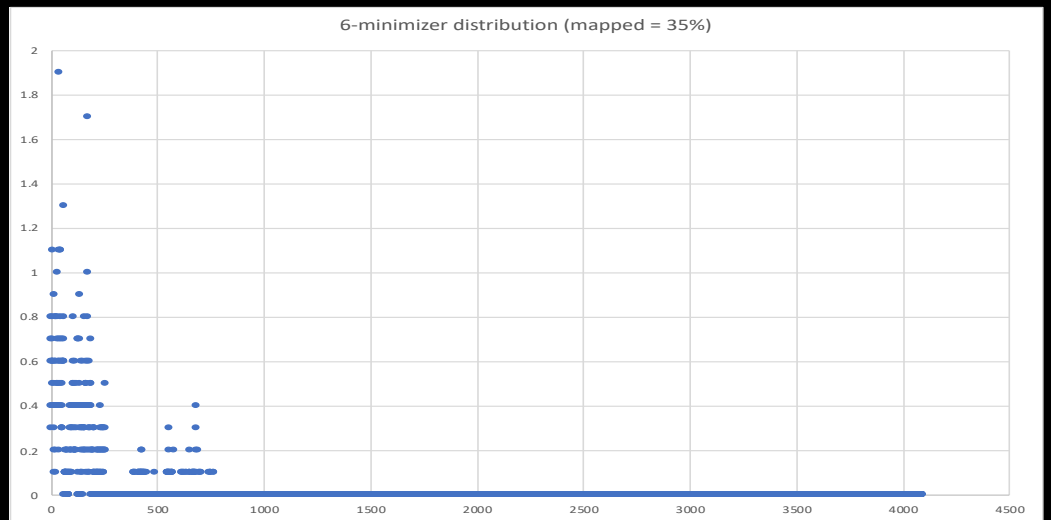
Tran[a] -> 2 (0.318858)
Tran[c] -> 1 (0.180059)
Tran[g] -> 0 (0.179884)
Tran[t] -> 3 (0.321199)

0 = gggg is the minimzer for
10.4% of 40-mers

2 = ggga is the minimzer for
12.8% of 40-mers







wellcome connecting science

m = 4

Largest is 12.2% => 7.8 parts

Supermer size = 17.3

m = 5

Largest is 4.7% => 21.2 parts

Supermer size = 16.6

m = 6

Largest is 1.9% => 53.5 parts

Supermer size = 16.1

Minimizer Skew & Bucketing

wellcome connecting science

50X Cabernet: 28Gbp ⇒ P = 28

40X Human: 120Gbp ⇒ P = 160

40X Axolotl: 1280Gbp ⇒ P = 2241

If need to split into P parts then must use

m(P) = min m such freq[w] ≤ 1/P for all m-minimizers w

where freq is empirically measured over a training data set.

m(P) can be quite big for real data

FastK has a novel way of avoiding an $O(4^{m(P)})$ mapping table

Was able to process 3Tbp Mistletoe data set in a 30 hour run.
I don't think the other counters out there can do this.

wellcome
connecting
science

FastK:

Looking at & Accessing the Results

Histex
Tabex
Profex

```
Interactive, we will run these on the results
  we produced in the first empirical session.
```

Histograms

wellcome connecting science

```
Histex [-kAG] [-h[<int(1)>:]<int(-G?1000:100)>] <source_root>[.hist]
```

By default a “pretty” print out from 1-100 (`-h1:100`).

```
> Histex CBS

Histogram of unique 40-mers of CBS

  Input: 2,037,951,741 unique 40-mers

     Freq:        Count   Cum. %
 >=   100:     14182086     0.7%
       99:       223761     0.7%
       98:       229929     0.7%
       97:       236805     0.7%
       96:       243810     0.7%
       95:       250450     0.8%
       94:       258153     0.8%
       . . .
        5:     12412925    29.3%
        4:     23282760    30.4%
        3:     51851727    33.0%
        2:    156442458    40.6%
        1:   1209760767   100.0%
```

Histograms

wellcome connecting science

```
Histex [-kAG] [-h[<int(1)>:]<int(-G?1000:100)>] <source_root>[.hist]
```

By default a “pretty” print out from 1-100 (`-h1:100`).

Can control the range with `-h` option, e.g. `-h3:80`

1 is the default low end, e.g. `-h50` = `-h1:50`

The top/bottom-most counts accumulate all below/above them.

```
> Histex CBS

Histogram of unique 40-mers of CBS

  Input: 2,037,951,741 unique 40-mers

     Freq:        Count    Cum. %
 >=   100:     14182086      0.7%
       99:       223761      0.7%
       98:       229929      0.7%
       97:       236805      0.7%
       96:       243810      0.7%
       95:       250450      0.8%
       94:       258153      0.8%
       . . .
        5:     12412925     29.3%
        4:     23282760     30.4%
        3:     51851727     33.0%
        2:    156442458     40.6%
        1:   1209760767    100.0%
```

```
> Histex -h3:80 CBS

Histogram of unique 40-mers of CBS

  Input: 2,037,951,741 unique 40-mers

     Freq:        Count    Cum. %
 >=    80:     20142610      1.0%
       79:       412424      1.0%
       78:       430676      1.0%
       77:       454828      1.1%
       76:       483451      1.1%
       75:       511324      1.1%
       74:       548932      1.1%
       . . .
        7:      5310951     28.3%
        6:      7626751     28.7%
        5:     12412925     29.3%
        4:     23282760     30.4%
 <=     3:   1418054952    100.0%
```

Clipped FASTK Histograms

wellcome connecting science

```
Histex -A -h1000 Ill | tail
   . . .
994     753
995     818
996     818
997     735
998     762
999     743
1000 699368
```

$= \sum_{f=1000}^{\infty} hist[f]$

$\Rightarrow \sum_{f=1}^{1000} hist[f] =$

$\sum_{f=1}^{\infty} hist[f] = 9{,}218{,}967{,}270$

The last (and first if ≠ 1) element of FastK histograms contains the sum of all counts above (and below) them

```
Histex -A -h10000 Ill | tail
   . . .
9994     11
9995      7
9996      6
9997      3
9998      4
9999     11
10000 63804
```

$= \sum_{f=10000}^{\infty} hist[f]$

$\Rightarrow \sum_{f=1}^{10000} hist[f] =$

$\sum_{f=1}^{\infty} hist[f] = 9{,}218{,}967{,}270$

```
Histex -A -h1 Ill | tail

     1 9218967270
```

Histograms

wellcome connecting science

```
Histex [-kAG] [-h[<int(1)>:]<int(-G?1000:100)>] <source_root>[.hist]
```

`-A` for tab-delimited ASCII

`-k` for counts of instances versus counts of unique k-mers

`-G` for Genomescope-ready histogram

```
> Histex CBS

Histogram of unique 40-mers of CBS

  Input: 2,037,951,741 unique 40-mers

     Freq:        Count    Cum. %
 >=   100:     14182086      0.7%
       99:       223761      0.7%
       98:       229929      0.7%
       97:       236805      0.7%
       96:       243810      0.7%
       95:       250450      0.8%
       94:       258153      0.8%
       . . .
        5:     12412925     29.3%
        4:     23282760     30.4%
        3:     51851727     33.0%
        2:    156442458     40.6%
        1:   1209760767    100.0%
```

```
> Histex -A CBS
1    1209760767
2    156442458
3    51851727
4    23282760
5    12412925
. . .
94   258153
95   250450
96   243810
97   236805
98   229929
99   223761
100  14182086
```

hist[k] = freq(k) = | a : count(a) = k |

```
> Histex -k CBS

Histogram of 40-mer instances of CBS

  Input: 27,890,711,413 40-mer instances

     Freq:        Count    Cum. %
 >=   100:   7877939960     28.2%
       99:     22152339     28.3%
       98:     22533042     28.4%
       97:     22970085     28.5%
       96:     23405760     28.6%
       95:     23792750     28.7%
       94:     24266382     28.7%
       . . .
        5:     62064625     93.6%
        4:     93131040     94.0%
        3:    155555181     94.5%
        2:    312884916     95.7%
        1:   1209760767    100.0%
```

hist[k] = k*freq(k)

Peaks are stronger

```
> Histex -G CBS
1    1209760767
2    156442458
3    51851727
4    23282760
5    12412925
. . .
994  814
995  882
996  873
997  857
998  825
999  881
1000 5194783
```

Value of `Histex -k -h1000 CBS`

GENOMESCOPE & FASTK

wellcome connecting science

```
FastK -v -k21 –t1 -T8 fArcCen1_S1_*.fastq.gz -NIll
      9:37u  58:11r+s   605%   Ill.ktab = 46.23GB  32G k-mers
Histex -G -h1000 Ill | GeneScopeFK -o GSO.Ill -k 21
```

len:1,029,515,401bp uniq:64.1%
aa:99.5% ab:0.526%
kcov:10.9 err:1.65% dup:0.854 k:21 p:2

GenomeScope2.0 requires an unbounded / complete histogram

```
Histex -A -h1000 Ill | GeneScope2.0
```

len:839,562,211bp uniq:78.6%
aa:99.5% ab:0.526%
kcov:10.9 err:1.96% dup:0.854 k:21 p:2

```
Histex -A -h10000 Ill | GeneScope2.0
```

len:937,384,385bp uniq:70.3%
aa:99.5% ab:0.526%
kcov:10.9 err:1.78% dup:0.854 k:21 p:2

GENOMESCOPE & FASTK

```
Histex -A -h1000 Ill | tail
   . . .
994     753
995     818
996     818
997     735
998     762
999     743
1000 699368
```

GenomeScope2.0 ignores last entry (GOOD!)

But uses $\sum_{f=1}^{1000} f*hist[f] = 27,680,503,714$ *as the total # of k-mers (BAD!)*

```
Histex -G -h1000 Ill | tail
   . . .
994          753
995          818
996          818
997          735
998          762
999          743
1000 4960546301
```

$= \sum_{f=1000}^{\infty} f*freq(f)$

$\Rightarrow \sum_{f=1}^{999} f*hist[f] + hist[1000]$

$= \sum_{f=1}^{\infty} f*freq(f) = 31,941,682,015$

and GeneScopeFK uses the last entry for genome size (PERFECT).

GENOMESCOPE & FASTK

wellcome connecting science

```
Histex -G -h1000 Ill | GeneScopeFK -o GSO.Ill -k 21
```

len:1,029,515,401bp uniq:64.1%
aa:99.5% ab:0.526%
kcov:10.9 err:1.65% dup:0.854 k:21 p:2

```
Histex -G -h10000 Ill | GeneScopeFK -o GSO.Ill -k 21
```

len:1,029,514,948bp uniq:64%
aa:99.5% ab:0.526%
kcov:10.9 err:1.65% dup:0.854 k:21 p:2

```
Histex -G -h200 Ill | GeneScopeFK -o GSO.Ill -k 21
```

len:1,029,515,365bp uniq:64.5%
aa:99.5% ab:0.526%
kcov:10.9 err:1.65% dup:0.854 k:21 p:2

Tables

wellcome connecting science

Tabex [-A] [-t<int>] <source_root>[.ktab] [LIST|CHECK|(k-mer:string>) ...)]

LIST = By default a “pretty” print out of the table

CHECK = ‘Table is OK’ or first k-mer out of order and its position

k-mer = either ‘Not Found’ or position and count of k-mer

```
> Tabex CBS LIST
Opening 40-mer table with 2,037,951,741 entries
          0: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa = 32767
          1: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac =  2530
          2: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaag =  3783
          3: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaat =  1414
          4: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaca =  1628
          5: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacc =   978
          6: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacg =   123
          7: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaact =   452
         . . .
 2037951731: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa =    10
 2037951732: ttttttttttttttttaattgaaaaaaaaaaaaaaaaaa =     1
 2037951733: ttttttttttttttttacttataaaaaaaaaaaaaaaaa =     3
 2037951734: ttttttttttttttttagaaaaaaaaaaaaaaaaaaaaa =     1
 2037951735: ttttttttttttttttcaattaaaaaaaaaaaaaaaaaa =     4
 2037951736: ttttttttttttttttcttgtaaaaaaaaaaaaaaaaaa =     1
 2037951737: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa =    18
 2037951738: ttttttttttttttttacaaaaaaaaaaaaaaaaaaaaa =     1
 2037951739: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa =    14
 2037951740: ttttttttttttttttacaaaaaaaaaaaaaaaaaaaaa =     4
```

```
> Tabex CBS Check
Opening 40-mer table with 2,037,951,741 entries
The table is OK
```

```
> Tabex CBS ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa
Opening 40-mer table with 2,037,951,741 entries
ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa:    10 @ idx = 2037951731
```

```
> Tabex CBS ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaac
Opening 40-mer table with 2,037,951,741 entries
ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaac: Not found
```

Tables

wellcome connecting science

```
Tabex [-A] [-t<int>] <source_root>[.ktab] [LIST|CHECK|(k-mer:string>) ...)]
```

Do not print k-mers whose count is less than -t option

```
> Tabex CBS LIST
Opening 40-mer table with 2,037,951,741 entries
          0: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa = 32767
          1: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac =  2530
          2: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaag =  3783
          3: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaat =  1414
          4: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaca =  1628
          5: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacc =   978
          6: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacg =   123
          7: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaact =   452
          8: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaga =  3231
          9: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagc =   157
         10: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagg =   656
        . . .
 2037951731: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaa =    10
 2037951732: ttttttttttttttttaattgaaaaaaaaaaaaaaaaa =     1
 2037951733: ttttttttttttttttacttataaaaaaaaaaaaaaaa =     3
 2037951734: ttttttttttttttttagaaaaaaaaaaaaaaaaaaaa =     1
 2037951735: ttttttttttttttttcaattaaaaaaaaaaaaaaaaa =     4
 2037951736: ttttttttttttttttcttgtaaaaaaaaaaaaaaaaa =     1
 2037951737: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaa =    18
 2037951738: ttttttttttttttttacaaaaaaaaaaaaaaaaaaaa =     1
 2037951739: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaa =    14
 2037951740: ttttttttttttttttacaaaaaaaaaaaaaaaaaaaa =     4
```

```
> Tabex -t200 CBS LIST
Opening 40-mer table with 2,037,951,741 entries
          0: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa = 32767
          1: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac =  2530
          2: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaag =  3783
          3: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaat =  1414
          4: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaca =  1628
          5: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacc =   978
          7: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaact =   452
          8: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaga =  3231
         10: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagg =   656
        . . .
 2037932344: ttttttttattttaaaaaatcccaaattaaaaaaaaaaaaa =   204
 2037934091: ttttttttcatattttttttttttttttttagaaaaaaaaaa =   270
 2037937943: ttttttttgaatttaaaatatacaaatagcatgaaaaaaaa =   818
 2037939305: ttttttttgtctttcctgaatgataaggagaaaaaaaaaaa =  1161
 2037941486: ttttttttaaatatatatattaaaaaaaaaaaaaaaaaaa =   306
 2037946518: ttttttttgtctttcctgaatgataaggagaaaaaaaaaa =   565
 2037947408: ttttttttttaaatatatatattaaaaaaaaaaaaaaaaa =   208
 2037948912: ttttttttttctcctcttgagttaagtaaaaaaaaaaaaa =  1031
 2037950461: ttttttttttctcctcttgagttaagtaaaaaaaaaaaaa =   659
 2037951347: ttttttttttttaaatatatatattaaaaaaaaaaaaaa =   210
```

Tables

wellcome connecting science

```
Tabex [-A] [-t<int>] <source_root>[.ktab] [LIST|CHECK|(k-mer:string>) ...)]
```

-A = table in tab-delimited form:

```
> Tabex CBS LIST
Opening 40-mer table with 2,037,951,741 entries
          0: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa = 32767
          1: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac =  2530
          2: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaag =  3783
          3: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaat =  1414
          4: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaca =  1628
          5: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacc =   978
          6: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacg =   123
          7: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaact =   452
          8: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaga =  3231
          9: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagc =   157
         10: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagg =   656
         . . .
 2037951731: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa =    10
 2037951732: ttttttttttttttttaattgaaaaaaaaaaaaaaaaaa =     1
 2037951733: ttttttttttttttttacttataaaaaaaaaaaaaaaaa =     3
 2037951734: ttttttttttttttttagaaaaaaaaaaaaaaaaaaaaa =     1
 2037951735: ttttttttttttttttcaattaaaaaaaaaaaaaaaaaa =     4
 2037951736: ttttttttttttttttcttgtaaaaaaaaaaaaaaaaaa =     1
 2037951737: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa =    18
 2037951738: ttttttttttttttttacaaaaaaaaaaaaaaaaaaaaa =     1
 2037951739: ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaaa =    14
 2037951740: ttttttttttttttttacaaaaaaaaaaaaaaaaaaaaa =     4
```

```
> Tabex -A CBS LIST
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa        32767
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac        2530
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaag        3783
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaat        1414
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaca        1628
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacc        978
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacg        123
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaact        452
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaga        3231
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagc        157
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagg        656
       . . .
ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaa        10
ttttttttttttttttaattgaaaaaaaaaaaaaaaa        1
ttttttttttttttttacttataaaaaaaaaaaaaaaa        3
ttttttttttttttttagaaaaaaaaaaaaaaaaaaa        1
ttttttttttttttttcaattaaaaaaaaaaaaaaaa        4
ttttttttttttttttcttgtaaaaaaaaaaaaaaaa        1
ttttttttttttttttgtaaaaaaaaaaaaaaaaaaa        18
ttttttttttttttttacaaaaaaaaaaaaaaaaaaa        1
ttttttttttttttttgtaaaaaaaaaaaaaaaaaaa        14
ttttttttttttttttacaaaaaaaaaaaaaaaaaaa        4
```

```
> Tabex -A CBS ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaac
ttttttttttttttttgtaaaaaaaaaaaaaaaaaaaa  10   2037951731
```

Profiles

```
Profex [-Az] <source_root>[.prof] [ <read:int>[-(<read:int>|#)] ... ]
```

Reads are 1 to # (denotes last read)

Can ask to see a list of individual reads or read ranges

```
Profex CBS 2
Read 2:
     0:    61
     1:    61
     2:    61
     3:    61
     4:    62
     5:    62
     6:    62
     7:    62
     8:    62
     9:    62
    10:    61
    11:    61
    12:    30
    13:    31
    14:    31
    15:    30
    16:    30
    17:    30
    18:    30
    19:    30
    20:    30
    21:    30
    22:    30
   . . .
 12224:    73
 12225:    69
 12226:   135
 12227:   135
 12228:    68
 12229:   201
 12230:    46
 12231:    47
 12232:    47
 12233:    47
 12234:    47
 12235:    47
 12236:    47
 12237:    47
 12238:    46
 12239:    46
 12240:    45
 12241:    45
 12242:    45
 12243:    46
>
```

wellcome connecting science

Profiles

```
Profex [-Az] <source_root>[.prof] [ <read:int>[-(<read:int>|#)] ... ]
```

Profile levels tend to stay the same,

so `-z` requests display runs:

```
> Profex -z CBS 2
Read 2:
     0 -     3 (61)
     4 -     9 (62)
    10 -    11 (61)
    12 -    12 (30)
    13 -    14 (31)
    15 -    22 (30)
    23 -    29 (31)
    30 -    46 (29)
    47 -    50 (28)
    51 -    51 (29)
    52 -    52 (60)
    53 -    67 (30)
    68 -    68 (31)
    69 -    70 (32)
    71 -    78 (31)
    79 -    83 (30)
    84 -   104 (31)
    . . .
 12225 - 12225 (69)
 12226 - 12227 (135)
 12228 - 12228 (68)
 12229 - 12229 (201)
 12230 - 12230 (46)
 12231 - 12237 (47)
 12238 - 12239 (46)
 12240 - 12242 (45)
 12243 - 12243 (46)
```

```
Profex CBS 2
Read 2:
     0:    61
     1:    61
     2:    61
     3:    61
     4:    62
     5:    62
     6:    62
     7:    62
     8:    62
     9:    62
    10:    61
    11:    61
    12:    30
    13:    31
    14:    31
    15:    30
    16:    30
    17:    30
    18:    30
    19:    30
    20:    30
    21:    30
    22:    30
    . . .
 12224:    73
 12225:    69
 12226:   135
 12227:   135
 12228:    68
 12229:   201
 12230:    46
 12231:    47
 12232:    47
 12233:    47
 12234:    47
 12235:    47
 12236:    47
 12237:    47
 12238:    46
 12239:    46
 12240:    45
 12241:    45
 12242:    45
 12243:    46
>
```

wellcome
connecting
science

Profiles

```
Profex [-Az] <source_root>[.prof] [ <read:int>[-(<read:int>|#)] ... ]
```

`-A` requests tab-delimited output

```
> Profex -z CBS 2
Read 2:
     0 -     3 (61)
     4 -     9 (62)
    10 -    11 (61)
    12 -    12 (30)
    13 -    14 (31)
    15 -    22 (30)
    23 -    29 (31)
    30 -    46 (29)
    47 -    50 (28)
    51 -    51 (29)
    52 -    52 (60)
    53 -    67 (30)
    68 -    68 (31)
    69 -    70 (32)
    71 -    78 (31)
    79 -    83 (30)
    84 -   104 (31)
   . . .
 12225 - 12225 (69)
 12226 - 12227 (135)
 12228 - 12228 (68)
 12229 - 12229 (201)
 12230 - 12230 (46)
 12231 - 12237 (47)
 12238 - 12239 (46)
 12240 - 12242 (45)
 12243 - 12243 (46)
```

```
> Profex -Az CBS 2
Read    2
0       3       61
4       9       62
10      11      61
12      12      30
13      14      31
15      22      30
23      29      31
30      46      29
47      50      28
51      51      29
52      52      60
53      67      30
68      68      31
69      70      32
71      78      31
79      83      30
84      104     31
. . .
```

```
Profex CBS 2
Read 2:
     0:    61
     1:    61
     2:    61
     3:    61
     4:    62
     5:    62
     6:    62
     7:    62
     8:    62
     9:    62
    10:    61
    11:    61
    12:    30
    13:    31
    14:    31
    15:    30
    16:    30
    17:    30
    18:    30
    19:    30
    20:    30
    21:    30
    22:    30
   . . .
 12224:    73
 12225:    69
 12226:   135
 12227:   135
 12228:    68
 12229:   201
 12230:    46
 12231:    47
 12232:    47
 12233:    47
 12234:    47
 12235:    47
 12236:    47
 12237:    47
 12238:    46
 12239:    46
 12240:    45
 12241:    45
 12242:    45
 12243:    46
>
```

```
Profex -A CBS 2
Read    2
61
61
61
61
62
62
62
62
62
62
61
61
30
31
31
30
30
30
30
30
30
. . .
```

wellcome connecting science

wellcome
connecting
science

FastK:

Logex: a Swiss Army knife for manipulating tables

Logical Combinations

wellcome connecting science

Each .ktab contains a set of k-mers and their counts.

Tables where k is the same, can be combined logically.

A | B = union(or) of A and B
= a table containing every k-mer that occurs in A or in B.

A & B = intersection(and) of A and B
= a table containing every k-mer that occurs in both A and B.

A - B = difference of A and B
= a table containing every k-mer that occurs in A but not B.

A ^ B = exclusive or of A and B
= a table containing every k-mer that occurs in A or B but not both.

Can have expressions of these, e.g. (A-B)|(B-A) or (A&B)-C

Precedence is & > ^ > - > |, e.g. A^B-C&D|E = ((A^B)-(C&D))|E

Logical (K-mer/Count) Combinations

wellcome connecting science

Each .ktab contains a set of k-mers and their counts.

A | B = union(or) of A and B

A & B = intersection(and) of A and B

A - B = difference of A and B

A ^ B = exclusive or of A and B

But every k-mer also has a count and so should the result.

Easy for ^ and - as each k-mer comes from either A or B and never both, so the count is that from A or B its source.

But what about the k-mers that are in both A and B when performing a union or intersection?

We add to the | or & operator symbols a count operator that determines what to do:

Logical (K-mer/Count) Combinations

wellcome connecting science

Each .ktab contains a set of k-mers and their counts.

A | B = union(or) of A and B

A & B = intersection(and) of A and B

We add to the | or & operator symbol a count operator that determines what to do:

A |+ B : take the sum of the counts

A |< B : take the minimum of the counts

A |> B : take the maximum of the counts

A |* B : take the average of the counts

A |. B : take the left count, the right count only when the left isn't present

A |- B : max(left count - right count,0) where count is 0 if not present

Same for &+, &<, …

K-mer Expressions

Filter Op's on Counts & GC-Content

wellcome connecting science

The postfix unary op [range] removes any k-mer whose count is not in its range:

A[5-10] = keep k-mers with count ∈ [5,10]

A[5-10,15-20,3] = keep k-mers with count ∈ [5-10] U [15,20] U [3]

A[-10,20-] = keep k-mers with count ∈ [1-10] U [20,∞]

e.g. (A |> B |> C |> D)[-3] : k-mers with count ≦ 3 in one of the 4 tables (and its min. count)

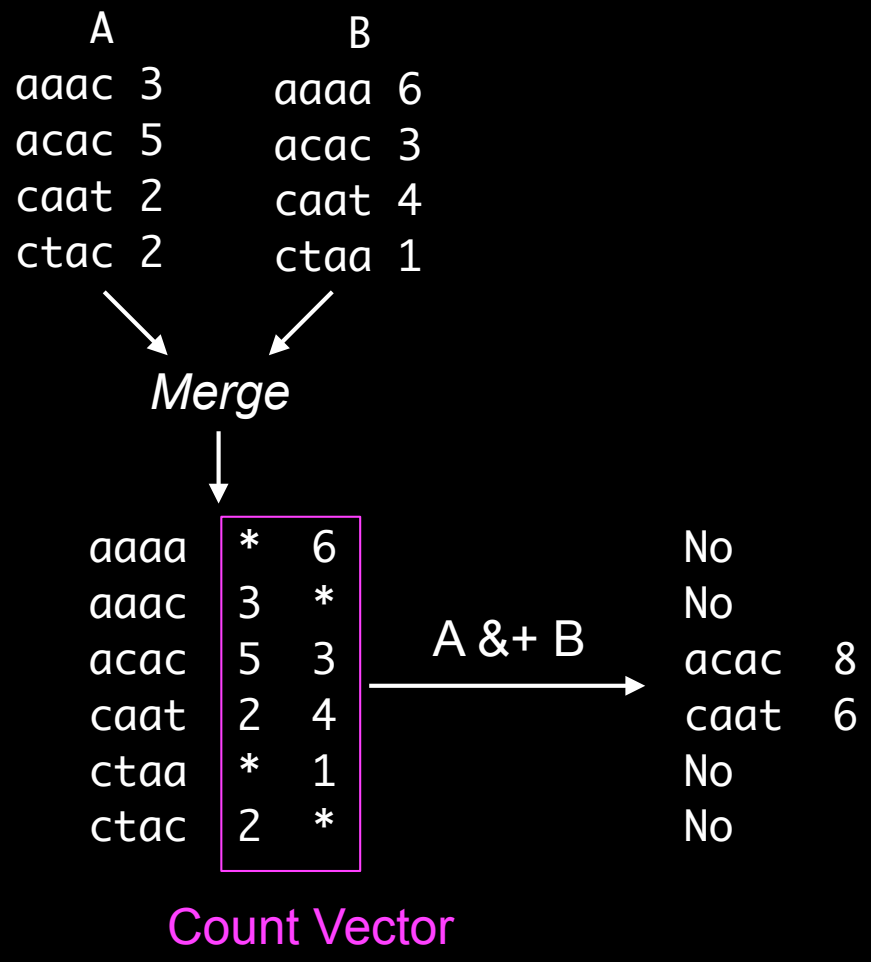
The postfix unary op {range} removes any k-mer whose GC percentage is not in its range. Exactly the same syntax as [], but integers must be between 0 and 100.

The prefix unary op # turns the counts of all k-mer to 1 !

e.g. #A |+ #B |+ #C : the count of any k-mer is the number of tables it is in, i.e. 1, 2, or 3.

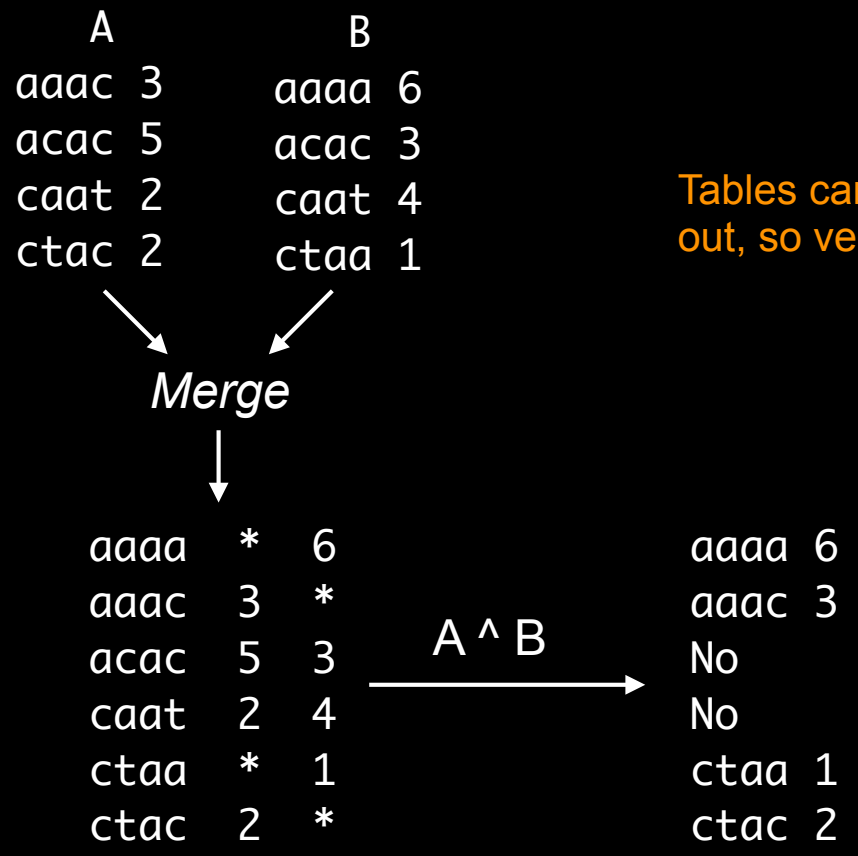
Logex Operation

K-mer expressions are evaluated by merging the input sorted k-mer tables in order of k-mer.



Logex Operation

K-mer expression are evaluated by merging the input sorted k-mer tables in order of k-mer.



Tables can be streamed in & result streamed out, so very little memory required.

Generalizes, an expression with n inputs is evaluated by an n-way merge of the tables evaluating the expression over each k-mer “count vector”.

Logex Operation:

1st input table
2nd input table
3rd input table

Logex ‘out=(A|+B)-C’ in1 in2 in3

out is the root name of output table, i.e. out.ktab

K-mer expression over 3 tables.
A is first input, B is second, and so on

Why not out=(in1|+in2)-in3 ?

Logex ‘out1=(A|+B)-C’ ‘out2=(B|+C)-A’ in1 in2 in3

Have to merge in1,2,&3 to compute out1 but also out2. So do it once and evaluate as many k-mer expressions over the inputs as you like. Felt it was simpler to map input names to A, B, C, …

Logex Finale:

wellcome connecting science

Logex [-T<int(4)>] [-[hH][<int(1)>:]<int(32767>] <name>=<expr> … <table> …

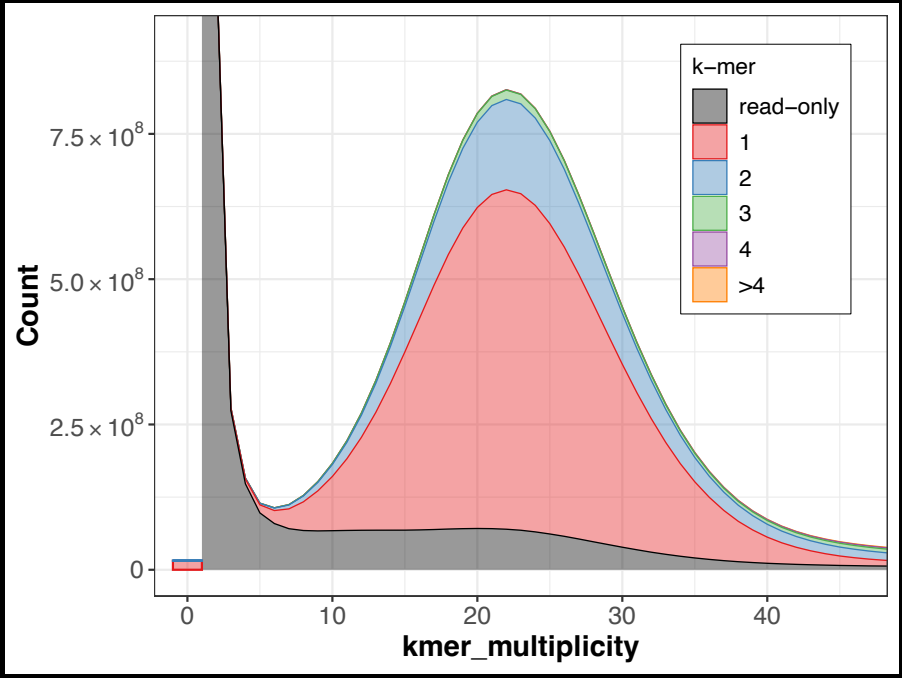
-T: How many threads to use.

-h: Produce a .hist also of each expression with given range

-H: Produce only a .hist and not a .ktab

Can have a maximum of 10 inputs

Logex Examples for Merqury (future session)



A Merqury CN-spectra plot

Look at spectra (histograms) of read k-mers separated according to the # of times they occur in an assembly of the reads.
coverage.

```
Logex -H1000 -T8 'H0=B-A' 'H1=B&.A[1]' … 'H4=B&.A[4]' 'H5m=B&.A[5-]' Asm Reads
```

Logex computes all 6 spectra in a single call that merges the read k-mer table with the assembly k-mer table only once!

All you want are the histogram so -H

Logex Examples for Merqury (future session)

wellcome connecting science

Given 2 haplotype phased assemblies Asm1 and Asm2 and a read data set of the individual, how much of the assemblies are covered by k-mers in the read data?

```
Logex -H1 -T8 'N1=A&.C[5-]' 'N2=B&.C[5-]' 'NU=(A|+B)&.C[5-]'
                                          'D1=A' 'D2=B' 'DU=A|>B' Asm1 Asm2 Reads
```

"Solid at 5 or above"

Number of k-mers!

Assembly	% Covered
Pri	97.83
Alt	97.07
Both	99.15

=N1.hist[1]/D1.hist[1]

Need to look at histogram of instances, not unique k-mers

wellcome
connecting
science

EXERICES

Some Logex-Solvable Problems

wellcome connecting science

Running Fastk in on an HPC cluster

If time permits, optional.

Job Template

wellcome connecting science

Want to split counting into multiple jobs to improve elapsed time.

Divide data set into N even parts and run FastK on each part producing just a table. (Profiles and histograms of a part are useless).

```
FastK -k32 -t Part1.fasta
FastK -k32 -t Part2.fasta
. . .
FastK -k32 -t PartN.fasta
```

Make sure -P refers to the local disk of the node the job is being run on! (and that it is large enought for the part)

Have now N k-mer tables, take their union.
Could use Logex if N ≦ 10, but not as efficient as:

```
Fastmerge -ht Full Part1 Part2 … PartN
```

Produce a table and a histogram

Getting Profiles

wellcome connecting science

If profiles are also desired then one has to run N more jobs:

```
FastK -k32 -p:Full Part1.fasta
FastK -k32 -p:Full Part2.fasta
. . .
FastK -k32 -p:Full PartN.fasta
```

Profile of part w.r.t. full k-mer table

And then these can be *concatenated* into a single result for all the parts with:

```
Fastcat -p Full Part1 Part2 … PartN
```

Concatenate the profiles

Certainly a lot more compute, but if the cluster isn't busy you should overall get it done faster with ~6 or more parts.

Avoiding Network IO:

wellcome connecting science

Fastmerge -ht Full Part1 Part2 … PartN

While the merge uses input buffers, it is still making a lot of read requests from the cluster's distributed file server.

If you specify a workspace with -P which should be the local disk of the job's node, then all of the parts are transferred there before performing the merge

Fastmerge -ht -P/scratch Full Part1 Part2 … PartN

This is OK as long as the local disk is large enough.

If not then you can divide each part into K "slices" so that each slice covers the same range of k-mer values, and merge each slice independently.

Fastmerge -ht -P/scratch -S1ofK Slice1 Part1 Part2 … PartN
Fastmerge -ht -P/scratch -S2ofK Slice2 Part1 Part2 … PartN
. . .
Fastmerge -ht -P/scratch -SKofK SliceK Part1 Part2 … PartN

Avoiding Network IO:

wellcome connecting science

The K slices can then be concatenated with Fastcat:

Fastcat -ht Full Slice1 Slice2 … SliceK

Profiles can then be produced from Full as before.

I do not know of anyone who has actually done this on an HPC cluster. The parts have all been tested individually. Seeing if it works as designed (and debugging any issues) would be "interesting" and I am curious as to performance.

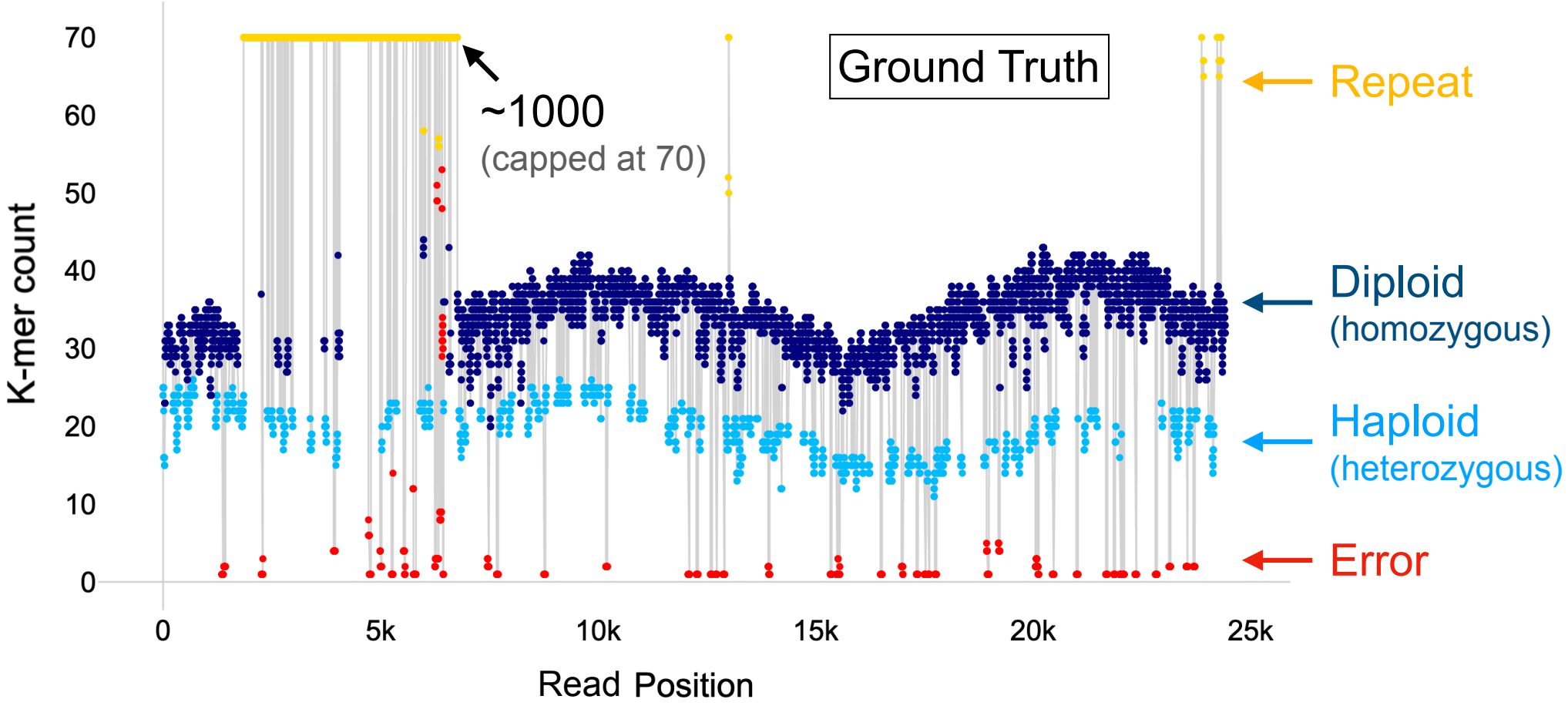
wellcome
connecting
science

A Profile-Based K-mer Classifier (A Preview)

w. Yoshihiko Suzuki

`If time permits, optional.`

ClassPro: K-mer classification based on read profiles

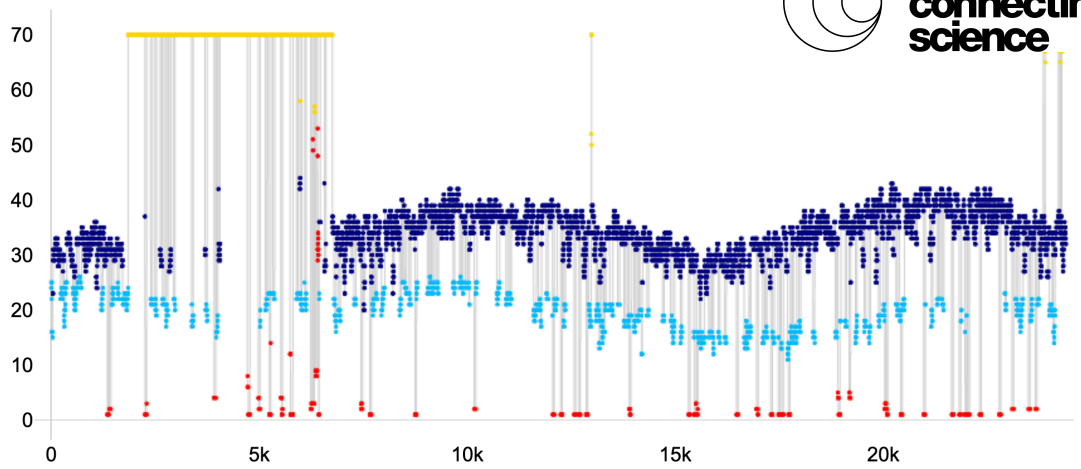


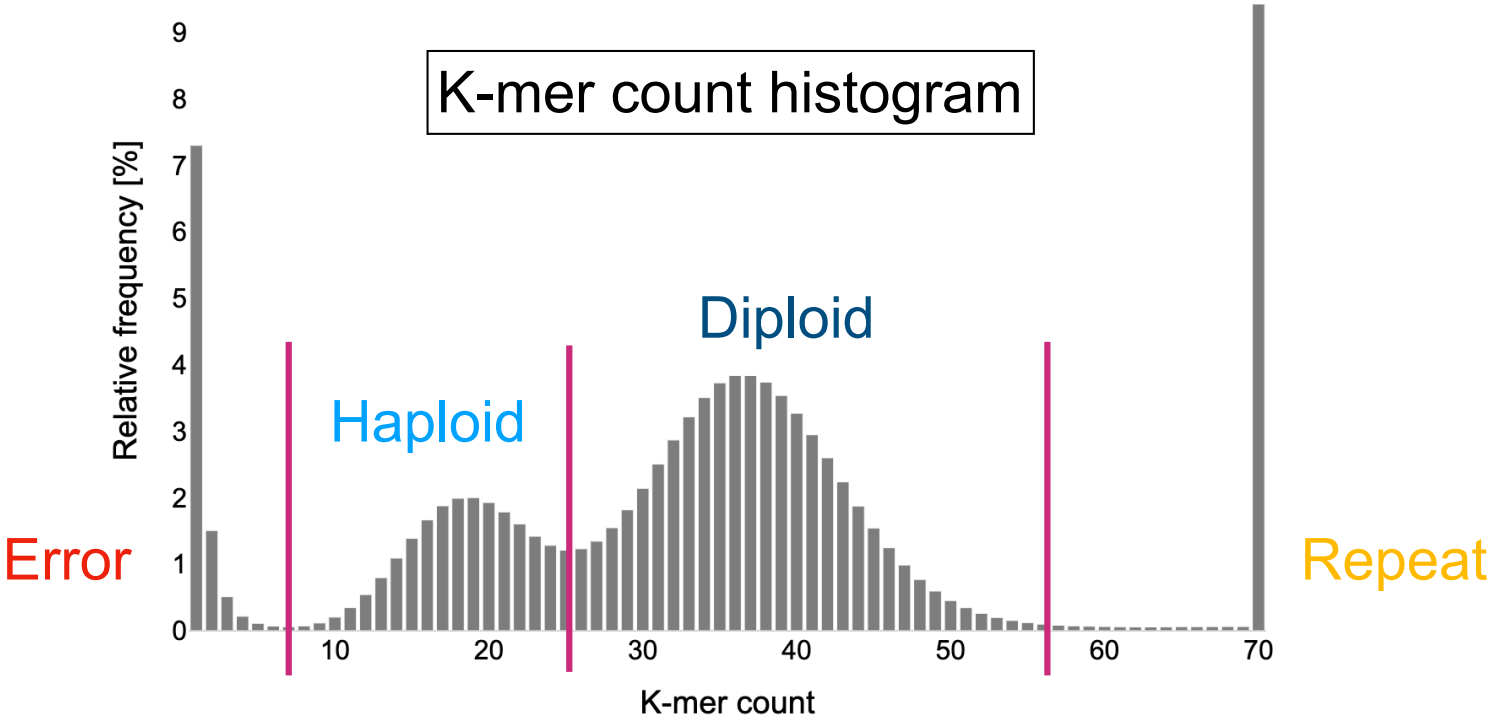
Dot = A k-mer ending at the position

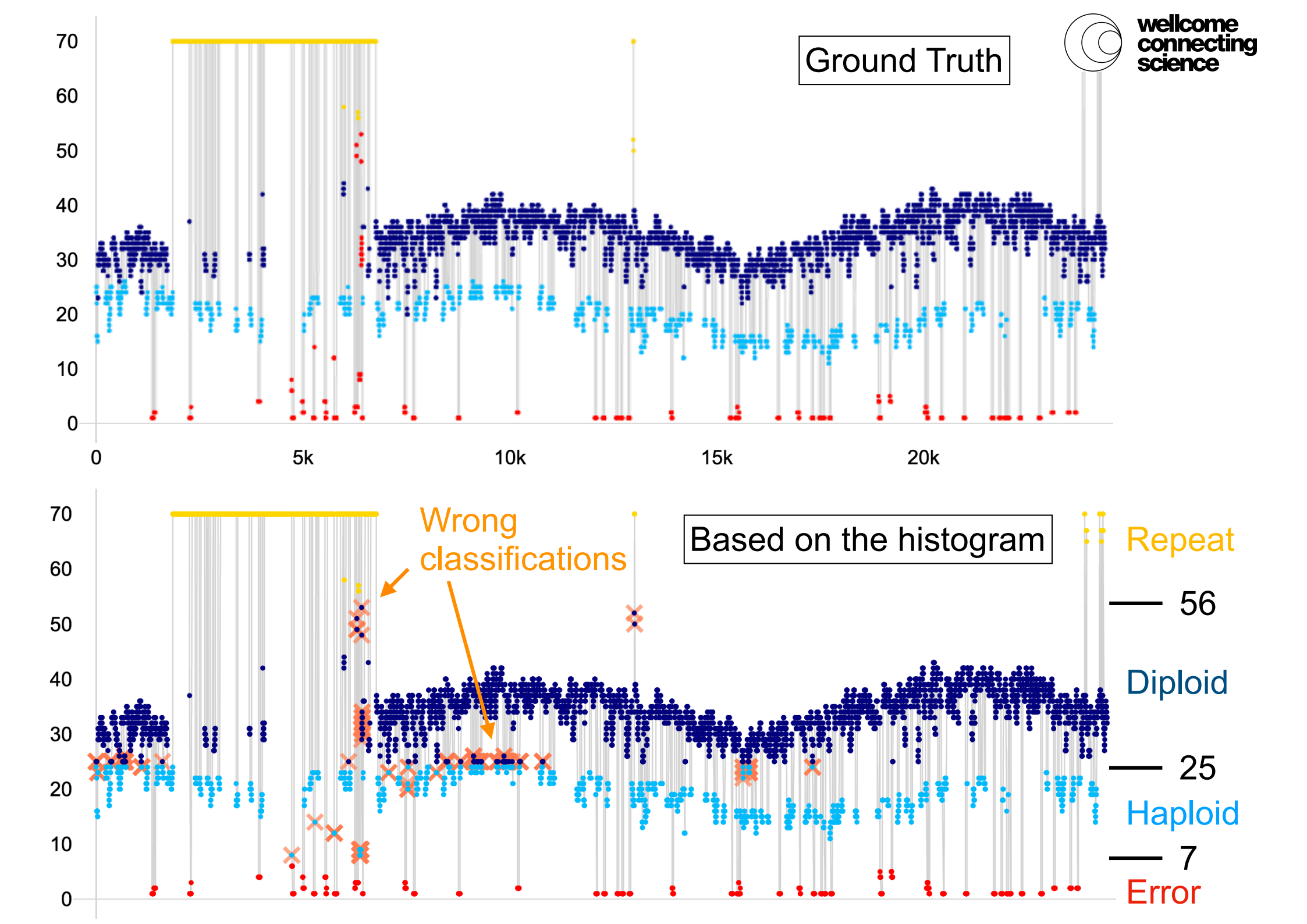
wellcome connecting science

(synthetic fly genome, ~0.7% het, 40x, K=40)

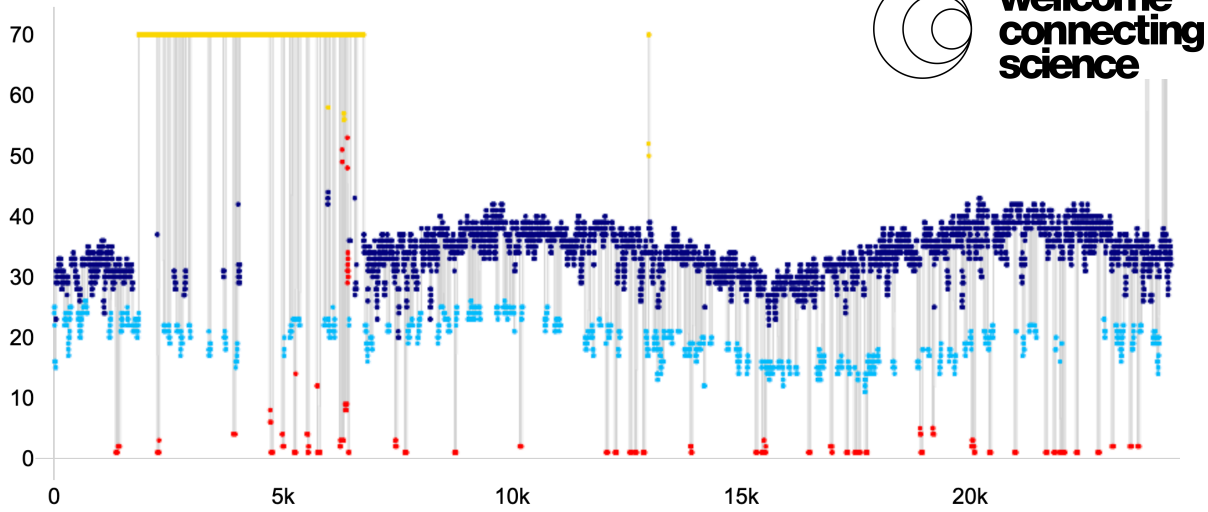
Approach to date:






Ground Truth
wellcome connecting science
70
60
50
40
30
20
10
0
0
5k
10k
15k
20k
Wrong classifications
Based on the histogram
Repeat
56
Diploid
25
Haploid
7
Error

Leveraging profiles



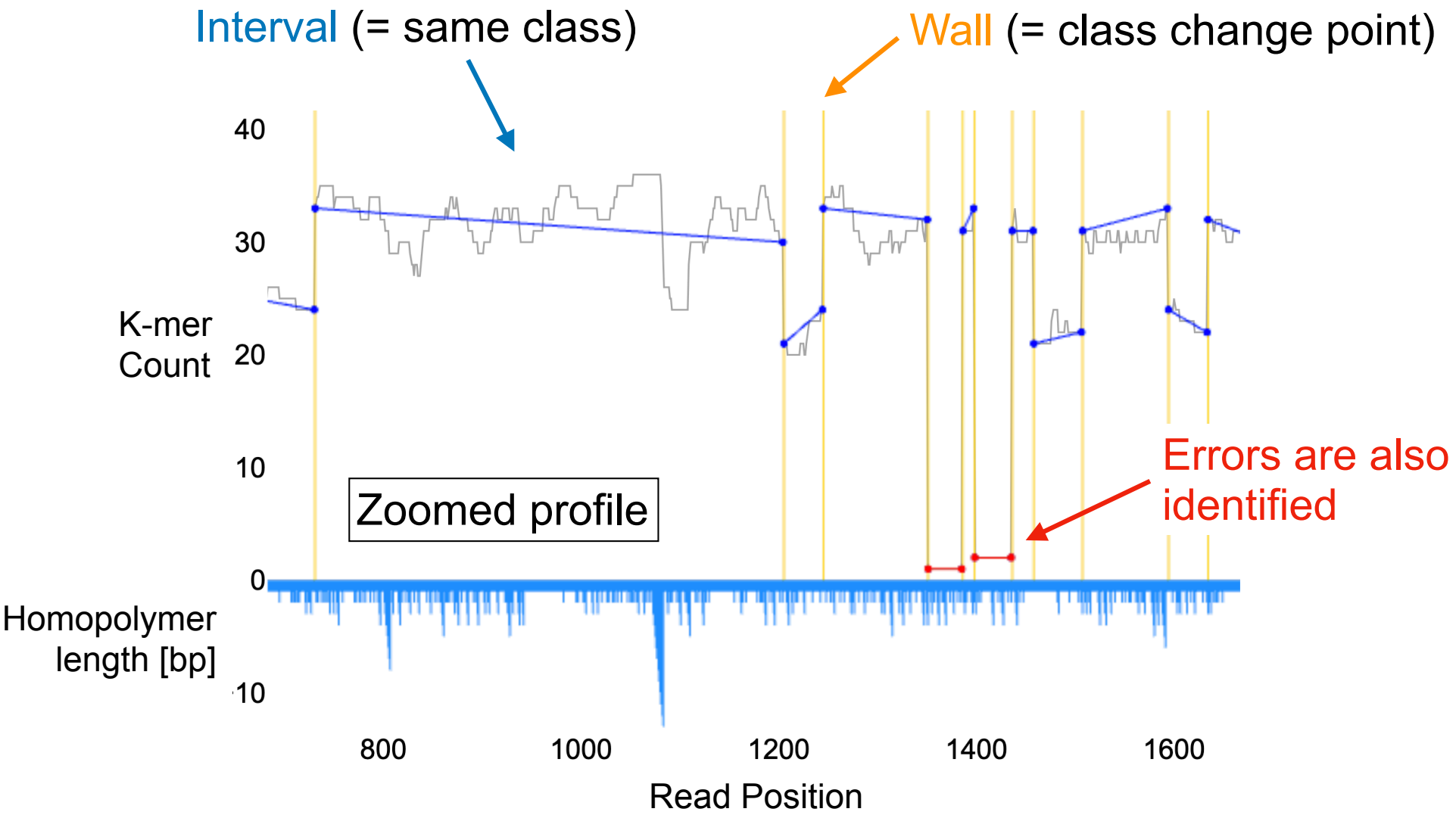
wellcome connecting science

1. **"Coherence"** principle
   - Adjacent k-mers with same class have similar counts
     - Fluctuation = 1) Poisson process of read arrival and exit
       2) Errors in reads other than the subject read

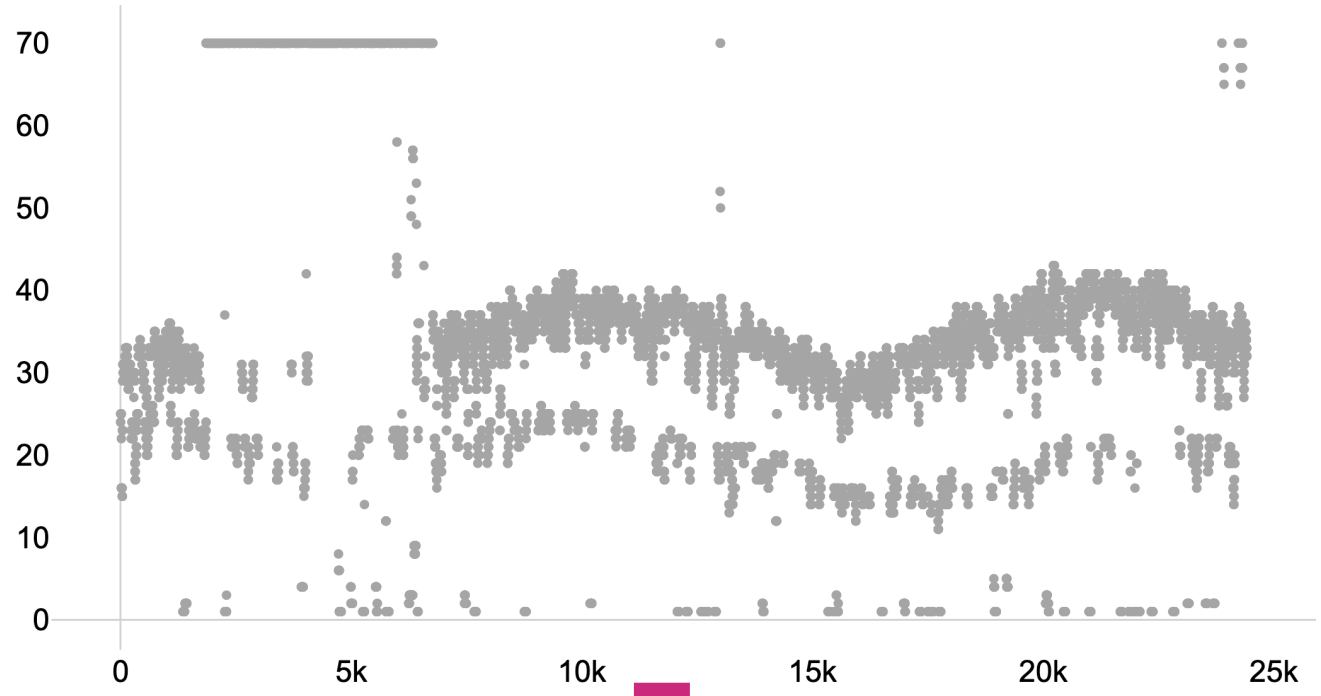
2. **"K-knockout"** principle
   - Transition to lower class lasts for ~K positions
     - e.g. Single error/variant affects ~K consecutive k-mers

3. Complication due to higher error rates of low-complexity sequences
   - e.g. Homopolymers like AAAA…

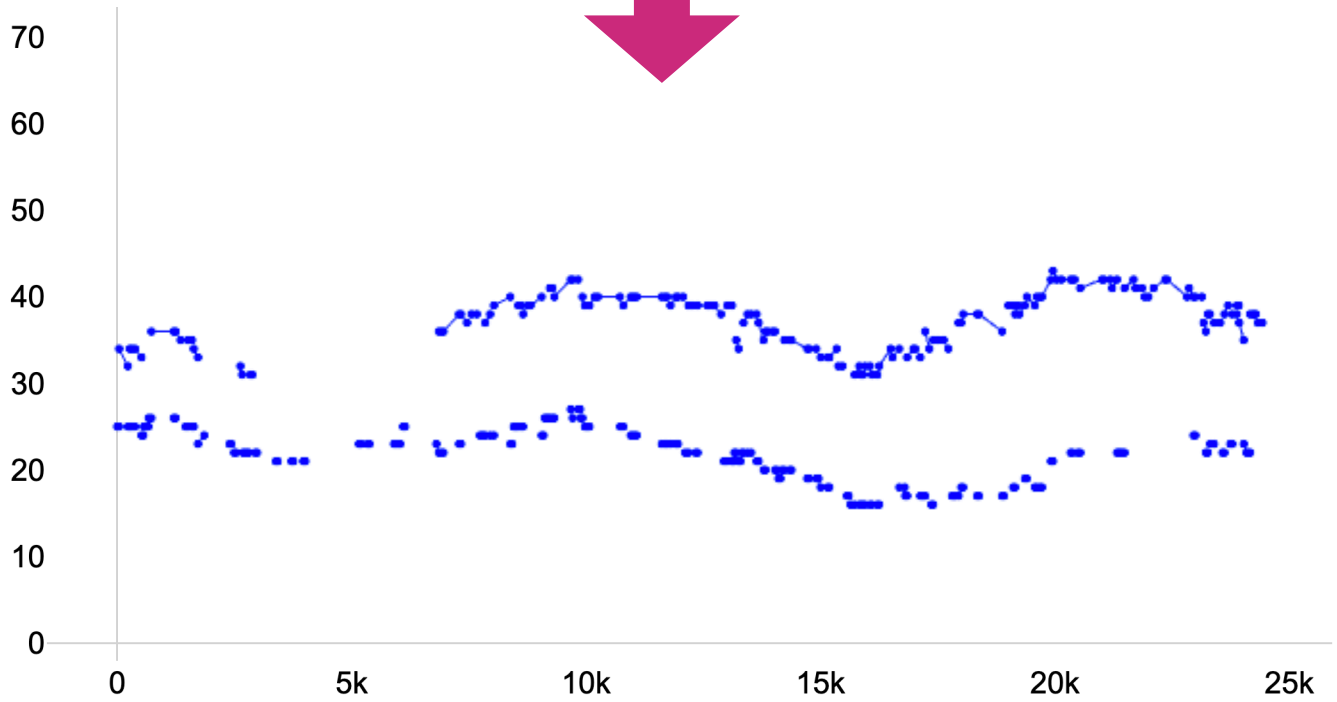
Algorithm Sketch: Walls and intervals



Algorithm Sketch: Walls and intervals

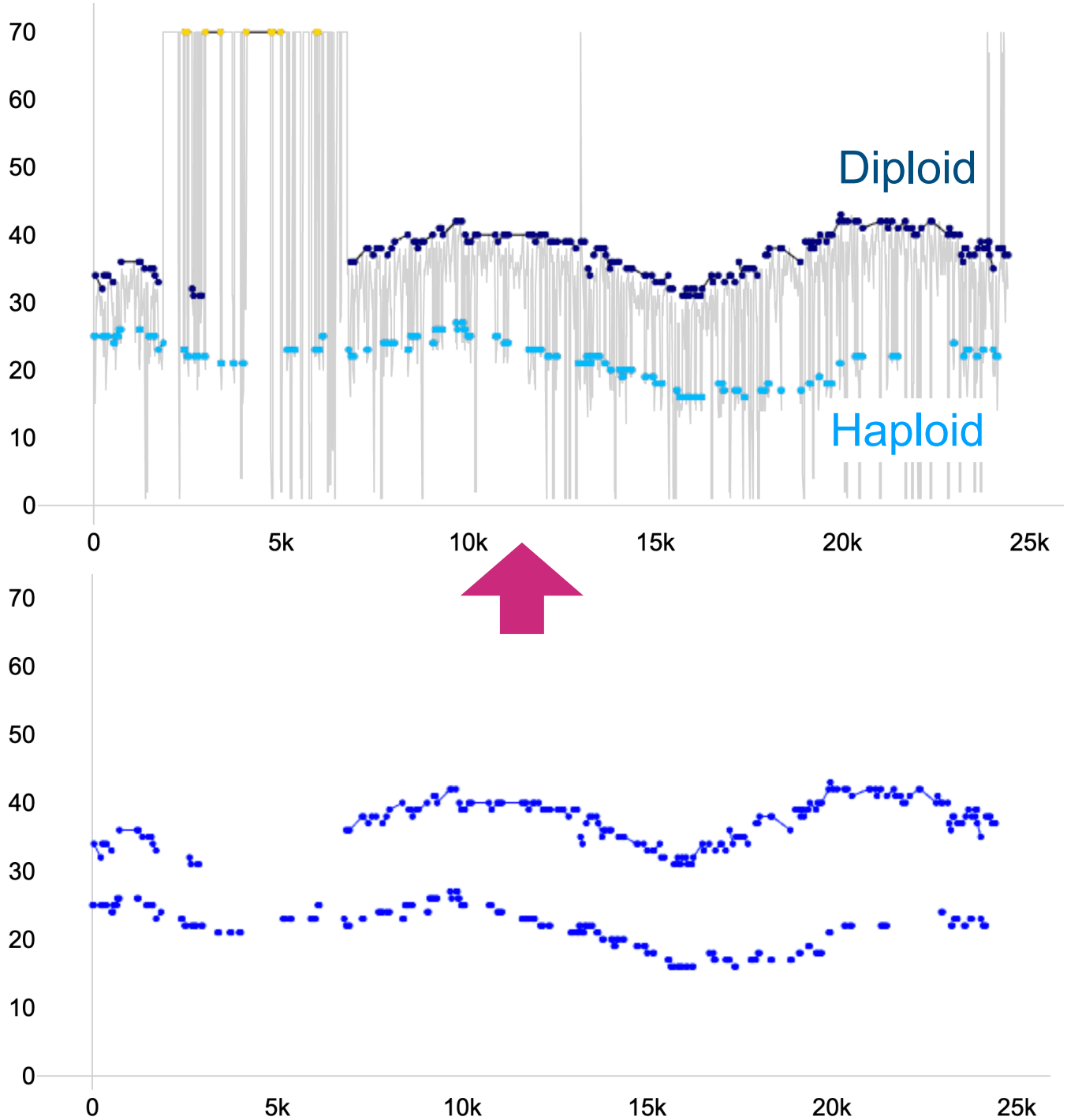


All K-mers



Long intervals with corrected wall counts

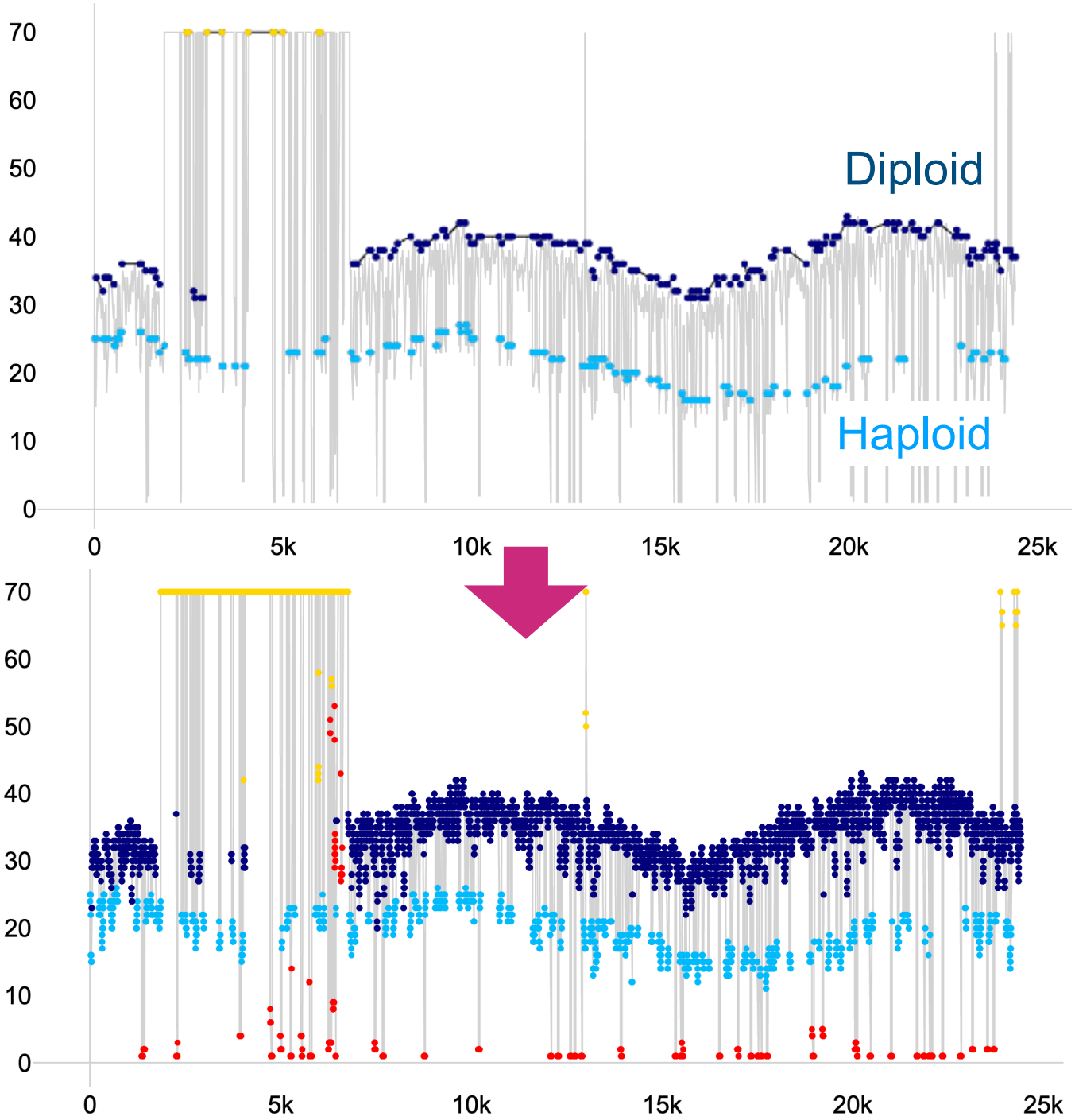
Algorithm Sketch: “Tier” finding



Identified backbone H/D-tiers

Long intervals with corrected wall counts

Algorithm Sketch: Filling the gaps



Identified backbone H/D-tiers

Final classifications

Performance on a simulation dataset

wellcome connecting science

(fruit fly diploid genome; ~0.7% het, 40x reads, K=40)

- Overall accuracy: **99.4%** (vs 95.6% based on histogram)
  - Real datasets have a larger read sampling fluctuation, making histogram-based classification even harder.
  - Accuracy for non-repetitive reads: **99.9%** (vs 96.4%)
- Accuracy for repetitive reads is weaker (98.0% (vs 93.7%)), but the false-negative rate of Error-mers is very low (0.5% (vs 1.5%)).
- ClassPro's classification for the same k-mer can be inconsistent among different reads (though overall consistency is **99.9%**).
  - Will offer an option to generate "consensus" classifications.
- https://github.com/yoshihikosuzuki/ClassPro (in beta)