

wellcome  
connecting  
science

# FastK Applications: MerquryFK, Smugeplot 2.0, KatFK

*Gene Myers*

*Sanger K-mer Class*  
*June 5, 2025*

# K-mer Counter Apps

*Counter:*

GenomeScope

*KMC*

SmudgePlot

*KMC*

Kat Tools

*JellyFish*

Merqury

*Meryl*

Merfin

*Meryl*

*Need to  
run/keep  
3 different  
counters/tables !*

# K-mer Counter Apps

github@thegenemyers

*FASTK*  
*MERQURY.FK*

## *Counter:*

GenomeScope

*FASTK*

*Need to*

SmudgePlot 2

*FASTK*

[Kamil+Gene]

Kat Tools

*FalseK*

[Only 3 different Merqury]

Merqury

*FastK*

*counters/tables !*

Merfin

*FastK*

- + *K-mer classification*
- + *Ab initio HiFi error model*

# The FastK “Ecosystem”

More efficient algorithms wherever possible

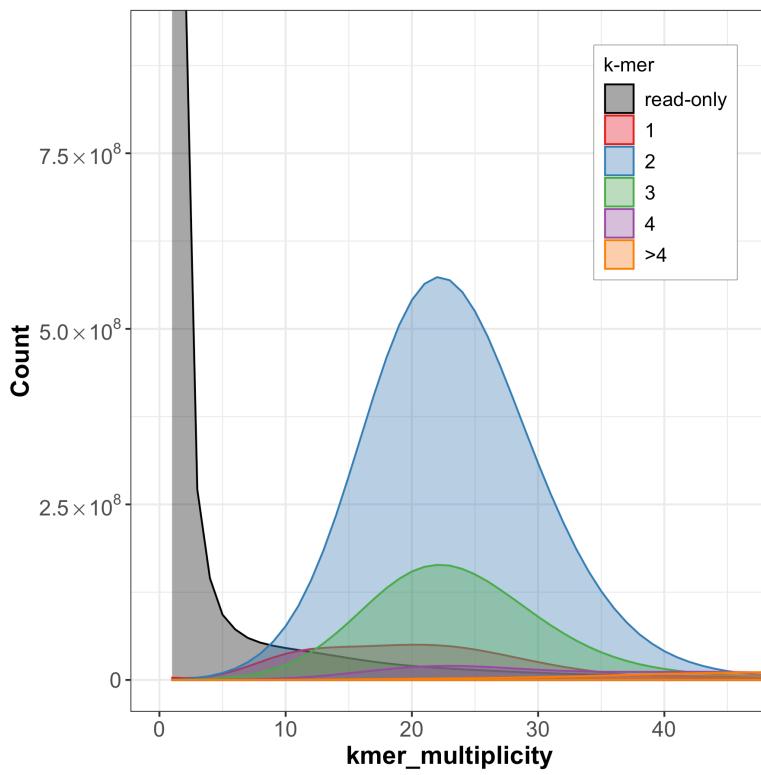
All apps are threaded

Tables never loaded, always efficiently streamed

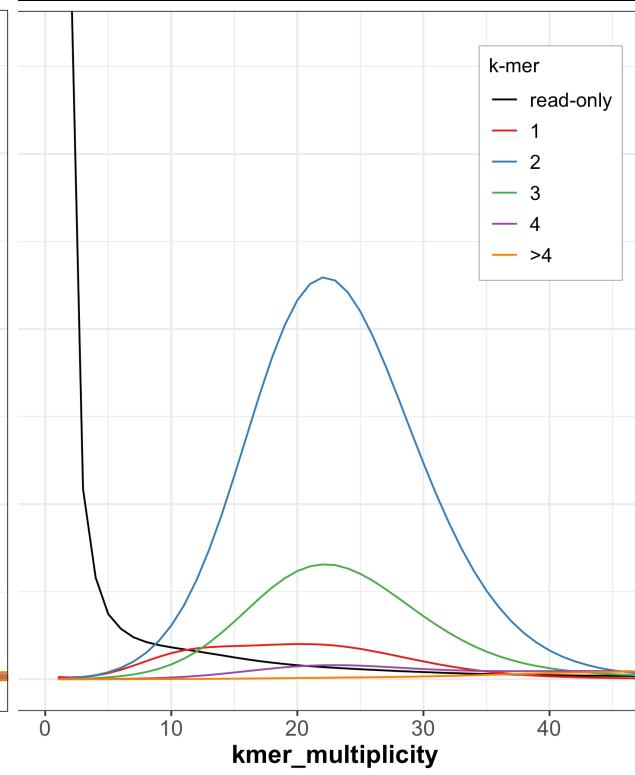
Common parameters conventions/style

Mercury  
Spectrum  
Plots

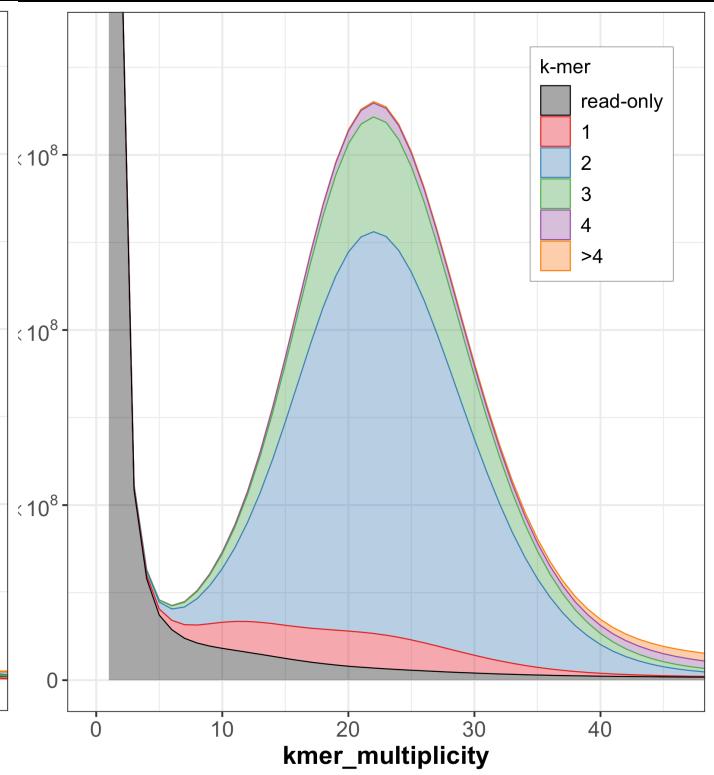
Fill, -f



Line, -l



Stack, -s



# The FastK “Ecosystem”

More efficient algorithms wherever possible

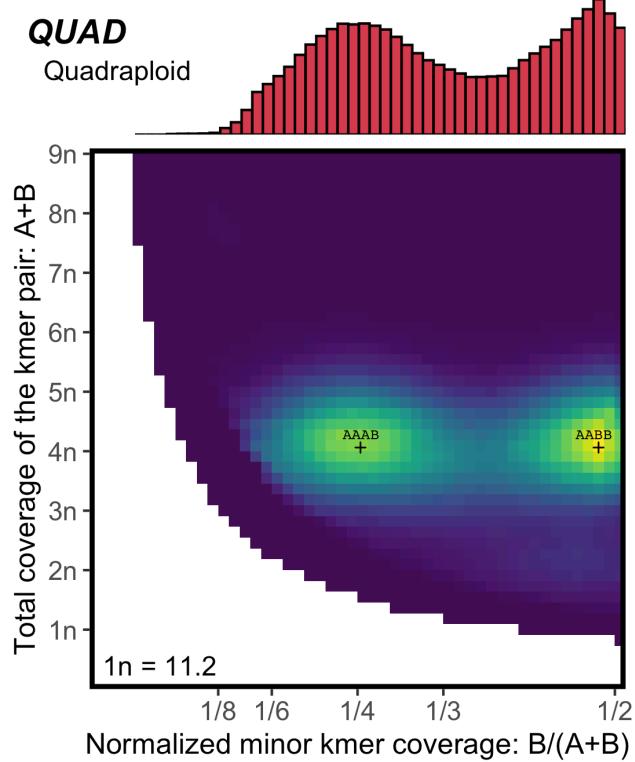
All apps are threaded

Tables never loaded, always efficiently streamed

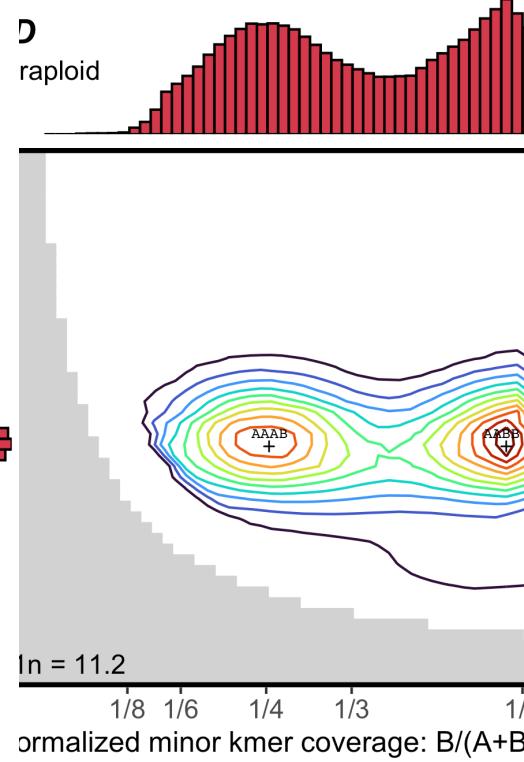
Common parameters conventions/style

PloidyPlot  
“smudge”  
Plots

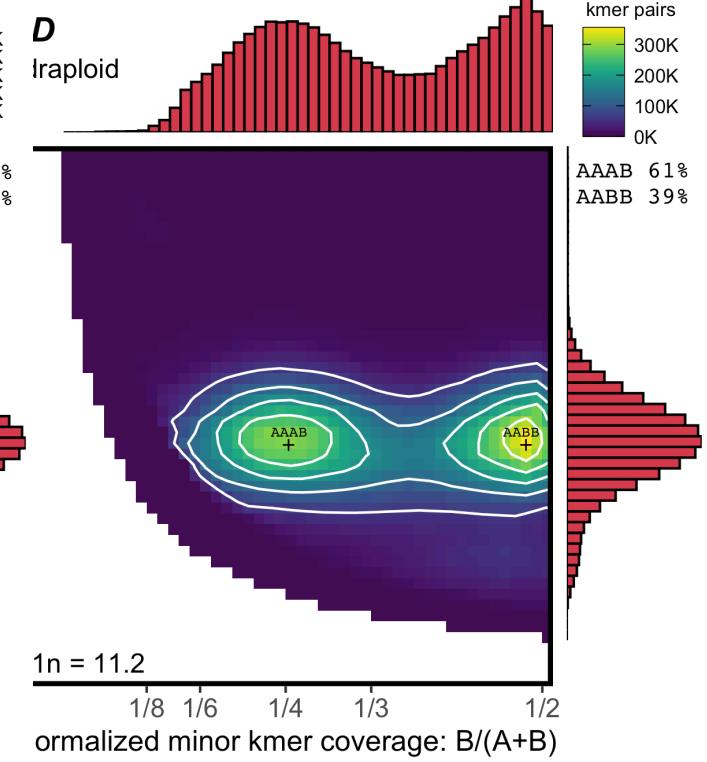
Fill, -f



Line, -l



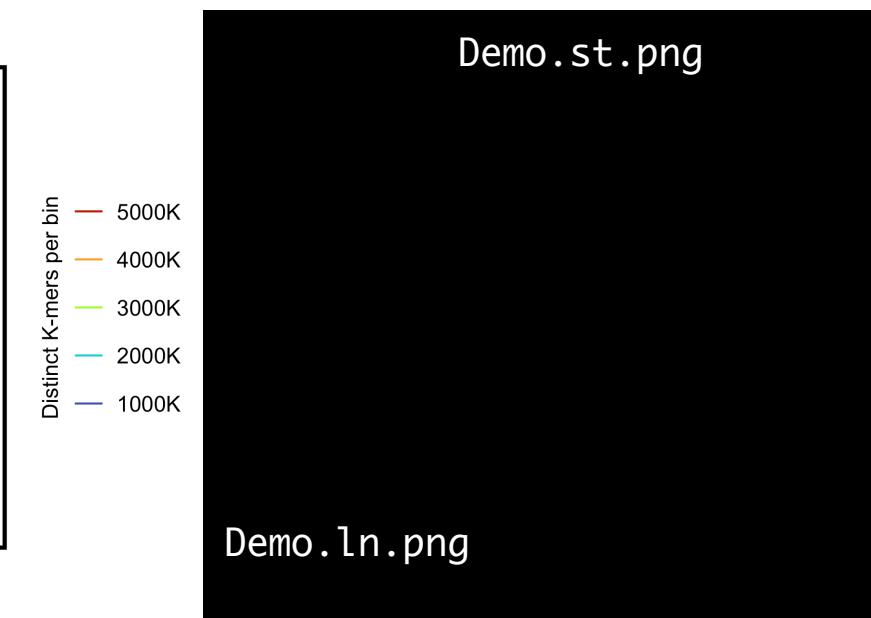
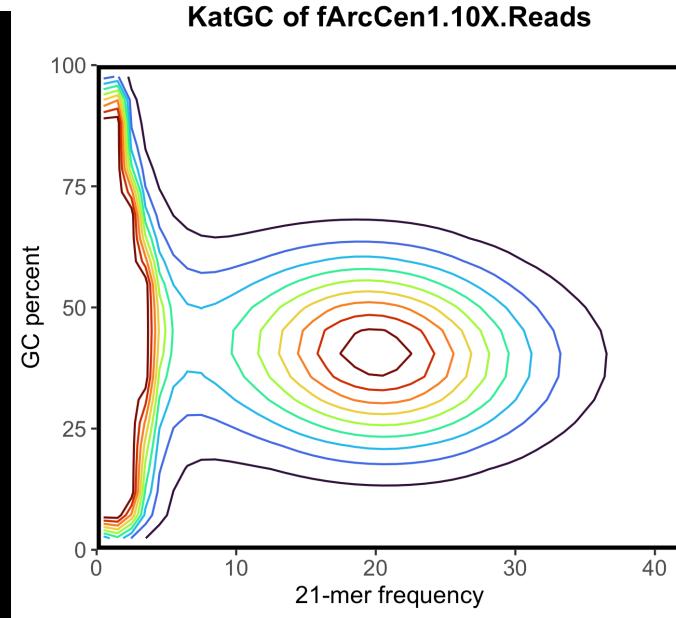
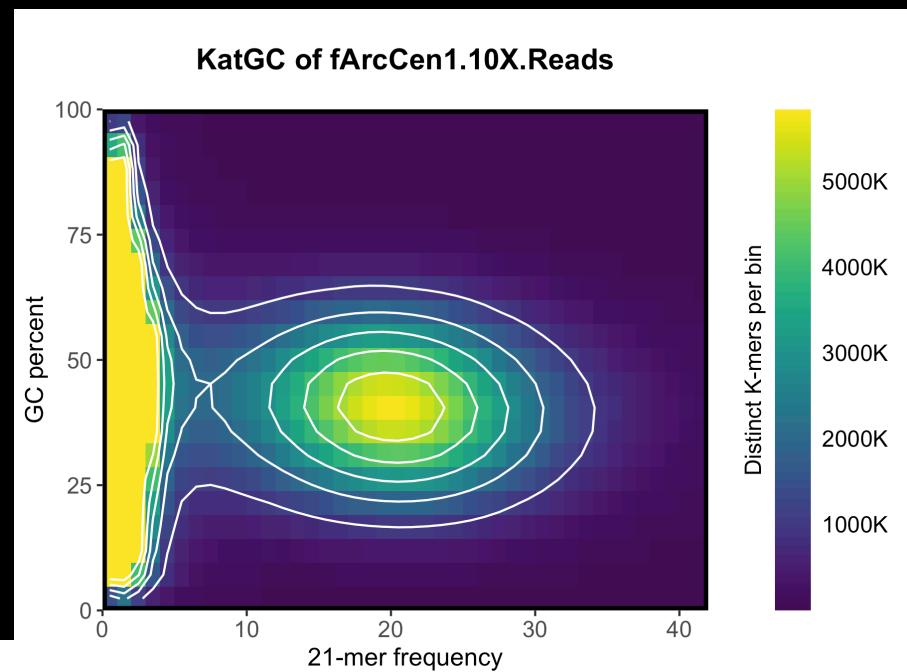
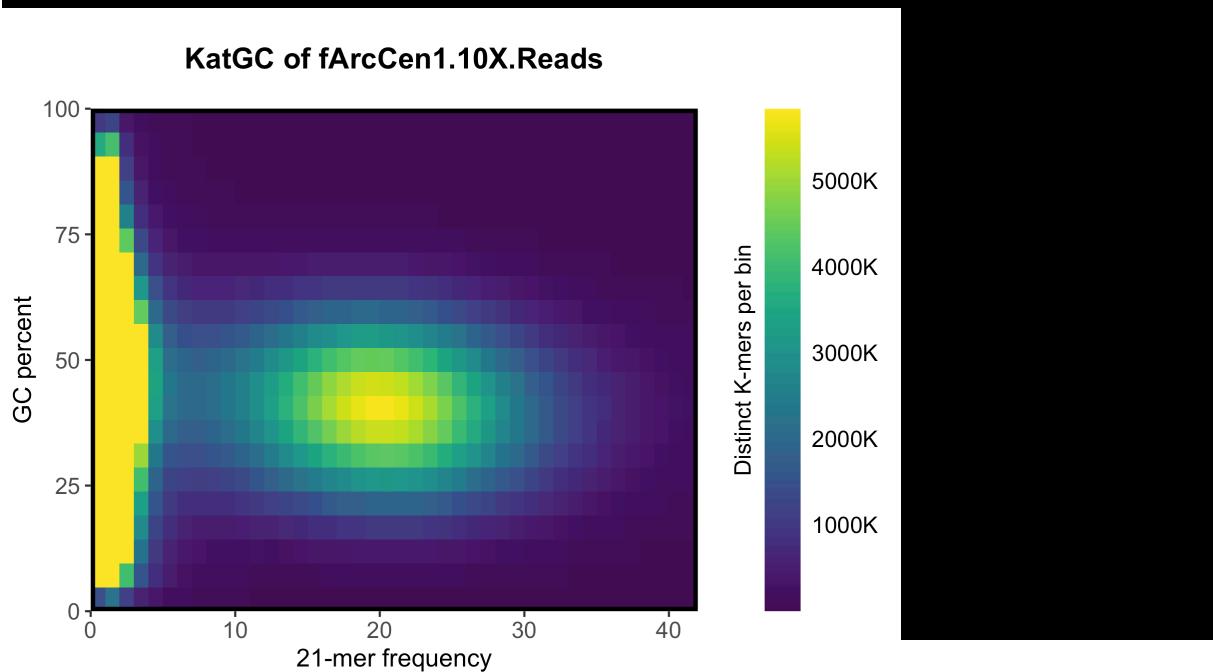
Stack, -s

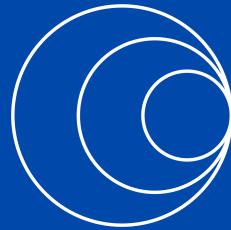


KatGC -T8 -oDemo fArcCen1.10X.Reads



25.5 seconds later ...





wellcome  
connecting  
science

MercuryFK

# MerquryFK

Merqury is a suite of k-mer analyses for QC'ing genome assemblies/reconstructions developed by Arang Rie.

The envisioned scenarios are one unphased assembly or two haplotype phased assemblies, with or without parental trio data:

**Asm1**: unphased assembly or one of two phased assemblies

**Asm2**: a second phased assembly

**Read**: read data from individual sequenced,  
e.g. Pacbio HiFi, Hi-C Illumina, or ONT ultra-long.

**Pat**: read data from individual's father

**Mat**: read data from individual's mother

Obviously designed for diploid genomes.

# Calling MerquryFK

So Merqury FK has 4 possible calls:

MerquryFK Read Asm1 Out (3 args)

MerquryFK Read Asm1 Asm2 Out (4 args)

MerquryFK Read Pat Mat Asm1 Out (5 args)

MerquryFK Read Pat Mat Asm1 Asm2 Out (6 args)

```
MerquryFK [-w<double(6.0)>] [-h<double(4.5)>]  
          [-[xX]<number(x2.1)>] [-[yY]<number(y1.1)>]  
          [-vk] [-lfs] [-pdf] [-z] [-T<int(4)>] [-P<dir(/tmp)>]  
          <read>[.ktab] [<mat>[.hap[.ktab]]] <pat>[.hap[.ktab]] ]  
          <asm1:dna> [<asm2:dna>] <out>
```

Note carefully the type of each argument!

# Setting up to Call MerquryFK

```
<read>[.ktab] [ <mat>[.hap[.ktab]] <pat>[.hap[.ktab]] ]  
<asm1:dna> [<asm2:dna>] <out>
```

The assemblies are sequence files (sam, bam, cram, fasta, fastq) where each sequence is a scaffold and gaps between contigs are denoted by runs of n's.

FastK needs to be run on the Read and Mat & Pat datasets if present. The Read's .ktab is the actual argument.

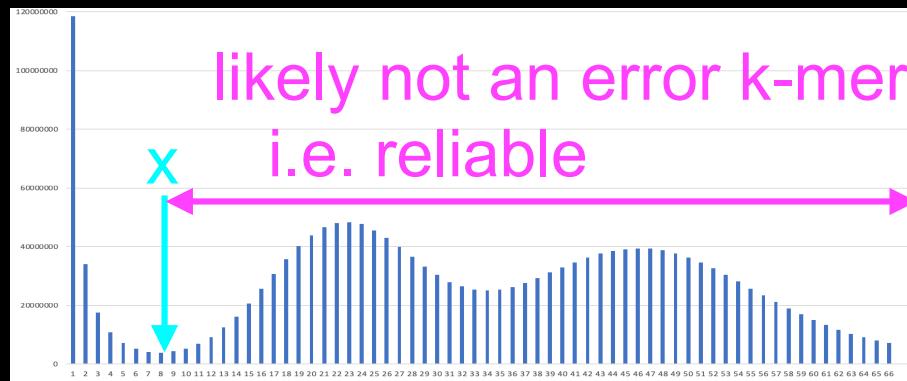
If trio data than HAPmaker needs to also be run beforehand to produce derived k-mer tables Mat.hap.ktab and Pat.hap.ktab, and these are the actual arguments.

# HAPmaker

```
HAPmaker [-v] [-T<int(4)>] <mat>[.ktab] <pat>[.ktab] <read>[.ktab]
```

$\text{Pat.hap} = \text{Rel}(\text{Read} \& \text{Rel}(\text{Pat-Mat}))$

$\text{Rel}(A) = A[x-]$  where  $x$  = 1st inflection in histogram of A



So, reliable k-mers in the father but not mother that are also reliable in the child.

$\text{Pat.hap} = \text{Rel}(\text{Read}) \& (\text{Rel}(\text{Pat})-\text{Mat})$  Just as good?

$\text{Mat.hap} = \text{Rel}(\text{Read} \& \text{Rel}(\text{Mat-Pat}))$

# MercuryFK Outputs

MercuryFK Read Pat Mat Asm1 Asm2 Out

Produces (lots of) outputs:

Out.Asm1.spectra-cn.fl.png  
Out.Asm1.spectra-cn.ln.png  
Out.Asm1.spectra-cn.st.png  
Out.Asm1.qv  
Out.Asm1\_only.bed

Out.spectra-asm.fl.png  
Out.spectra-asm.ln.png  
Out.spectra-asm.st.png  
Out.qv  
Out.completeness.stats

Out.Asm2.spectra-cn.fl.png  
Out.Asm2.spectra-cn.ln.png  
Out.Asm2.spectra-cn.st.png  
Out.Asm2.qv  
Out.Asm2\_only.bed

Out.spectra-cn.fl.png  
Out.spectra-cn.ln.png  
Out.spectra-cn.st.png

Out.Asm1.Mat.spectra-cn.fl.png  
Out.Asm1.Mat.spectra-cn.ln.png  
Out.Asm1.Mat.spectra-cn.st.png  
Out.Asm1.Pat.spectra-cn.fl.png  
Out.Asm1.Pat.spectra-cn.ln.png  
Out.Asm1.Pat.spectra-cn.st.png  
Out.Asm1.phased\_block.bed  
Out.Asm1.phased\_block.blob.png  
Out.Asm1.phased\_block.stats  
Out.Asm1.block.N.png  
Out.Asm1.continuity.N.png

Out.hapmers.blob.png

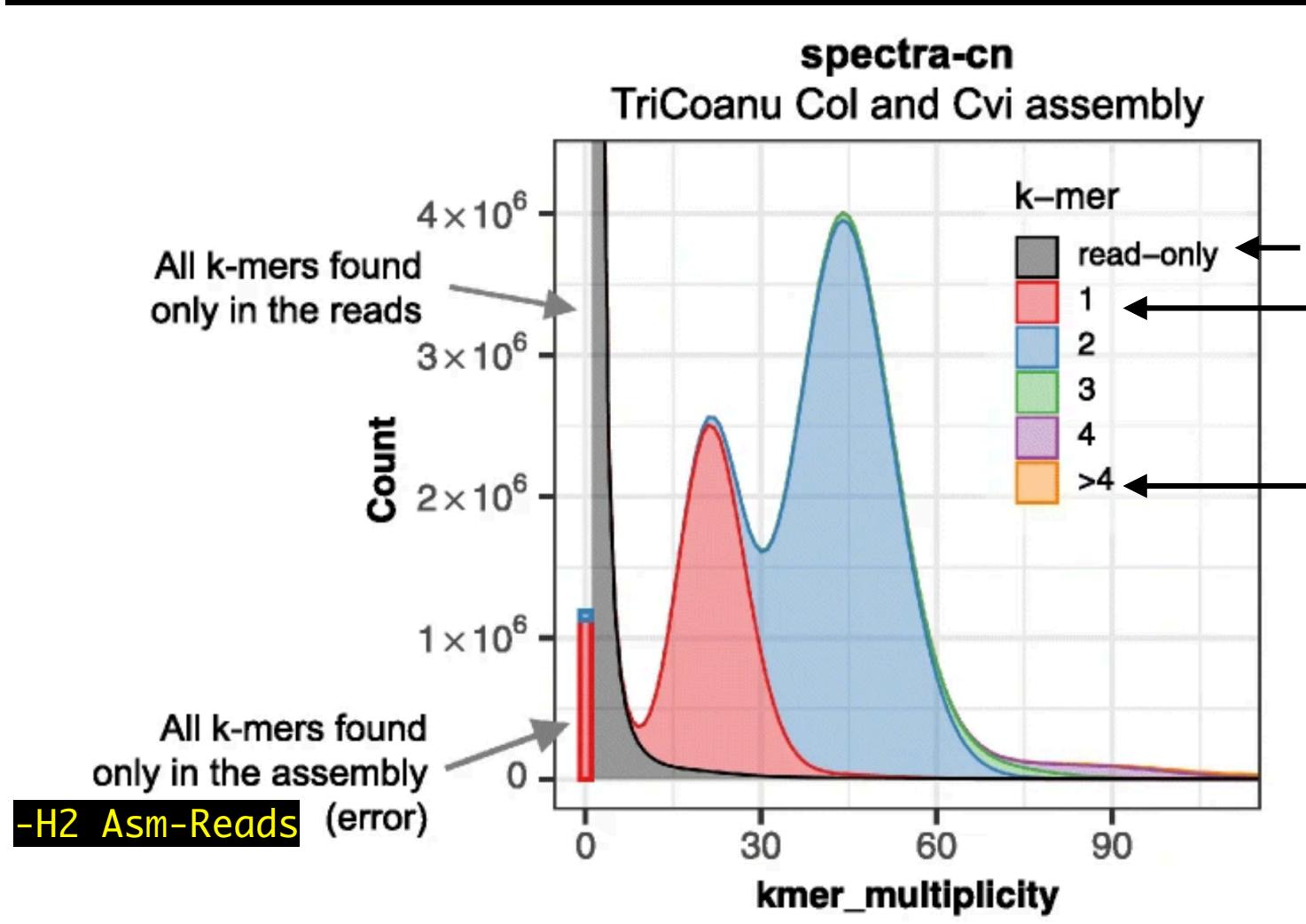
Out.Asm2.Mat.spectra-cn.fl.png  
Out.Asm2.Mat.spectra-cn.ln.png  
Out.Asm2.Mat.spectra-cn.st.png  
Out.Asm2.Pat.spectra-cn.fl.png  
Out.Asm2.Pat.spectra-cn.ln.png  
Out.Asm2.Pat.spectra-cn.st.png  
Out.Asm2.phased\_block.bed  
Out.Asm2.phased\_block.blob.png  
Out.Asm2.phased\_block.stats  
Out.Asm2.block.N.png  
Out.Asm2.continuity.N.png

We'll go through them one by one ...

# Copy Number Spectra Plots:

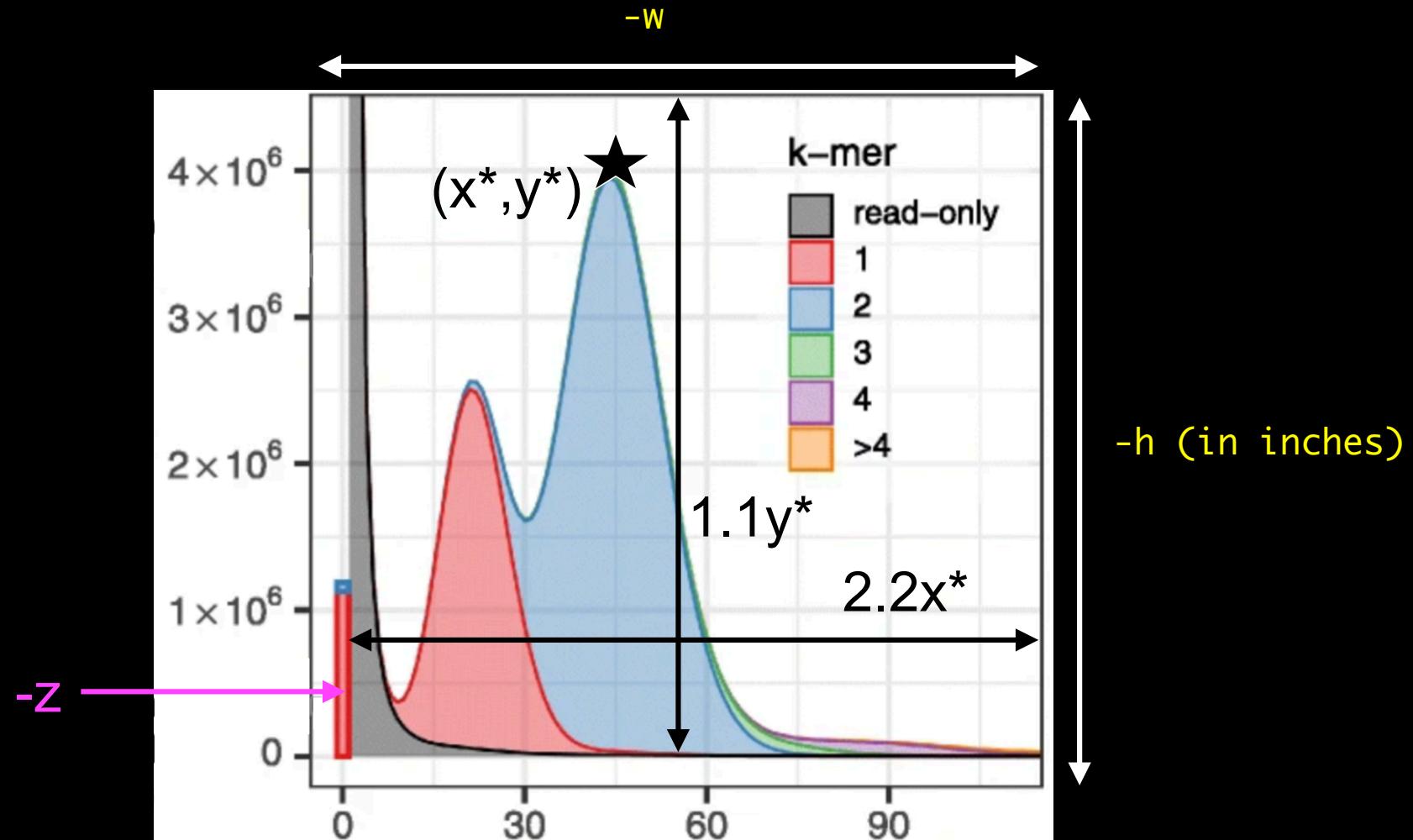
```
CNplot <reads>[.ktab] <asm:dna> <out>[.cni]
```

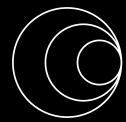
The output(s) is(are) <out>.[ln|fl|st].[png|pdf]



# CN-Spectra Options:

```
CNplot [-w<double(6.0)>] [-h<double(4.5)>] [-z] [-lfs]
[-[xX]<number(x2.1)>] [-[yY]<number(y1.1)>] [-pdf]
<reads>[.ktab] <asm:dna> <out>[.cni]
```





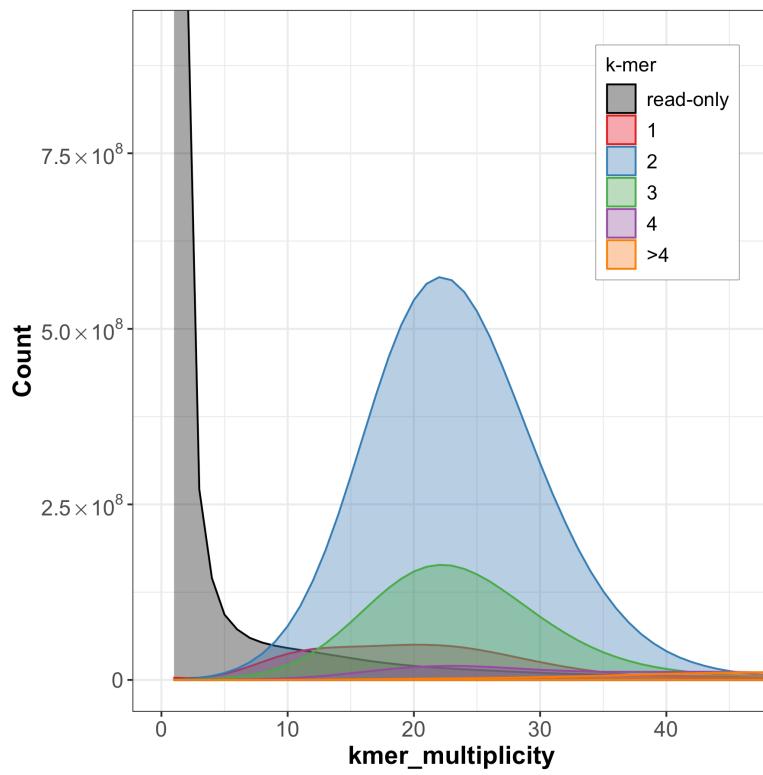
# CN-Spectra Options:

default is all 3  
e.g. -ls legal

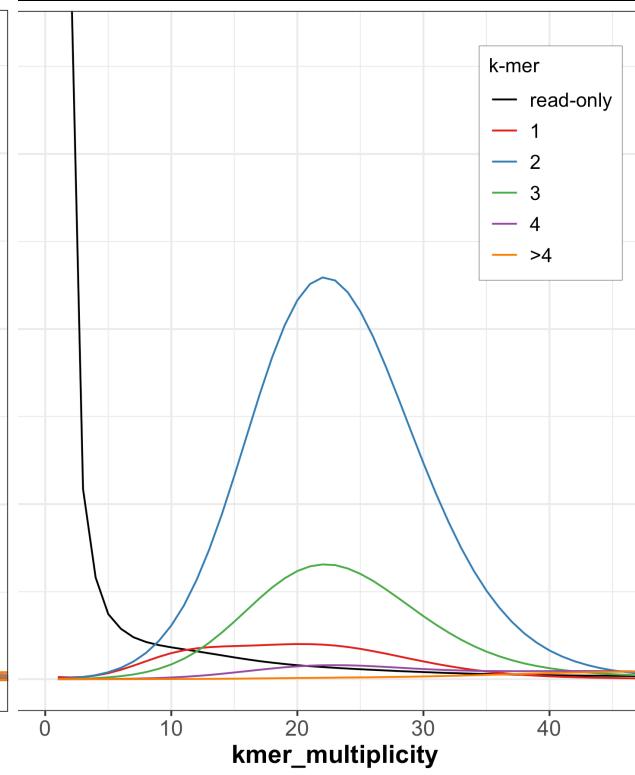
```
CNplot [-w<double(6.0)>] [-h<double(4.5)>] [-z] [-lfs]  
[-[xX]<number(x2.1)>] [-[yY]<number(y1.1)>] [-pdf]  
<reads>[.ktab] <asm:dna> <out>[.cni]
```

default is .png

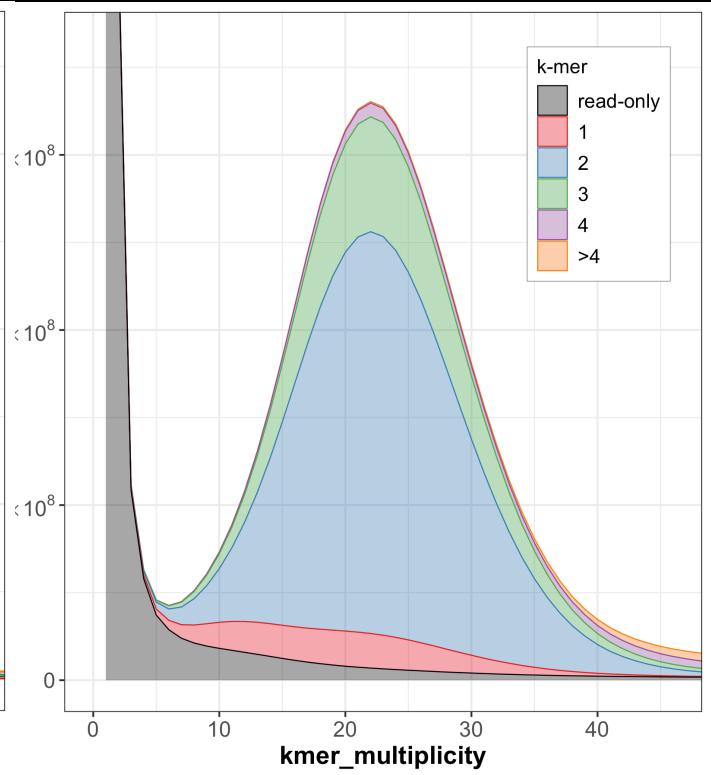
Fill, -f



Line, -l



Stack, -s



# CN-Spectra Options:

```
CNplot [-w<double(6.0)>] [-h<double(4.5)>] [-z] [-lfs]
[-[xX]<number(x2.1)>] [-[yY]<number(y1.1)>] [-pdf]
[-vk] [-T<int(4)>] [-P<dir(/tmp)>] ← Passed through to
<reads>[.ktab] <asm:dna> <out>[.cni] FastK and Logex
```

The **-k** option asks CNplot to save the FastK & Logex results so that a call:

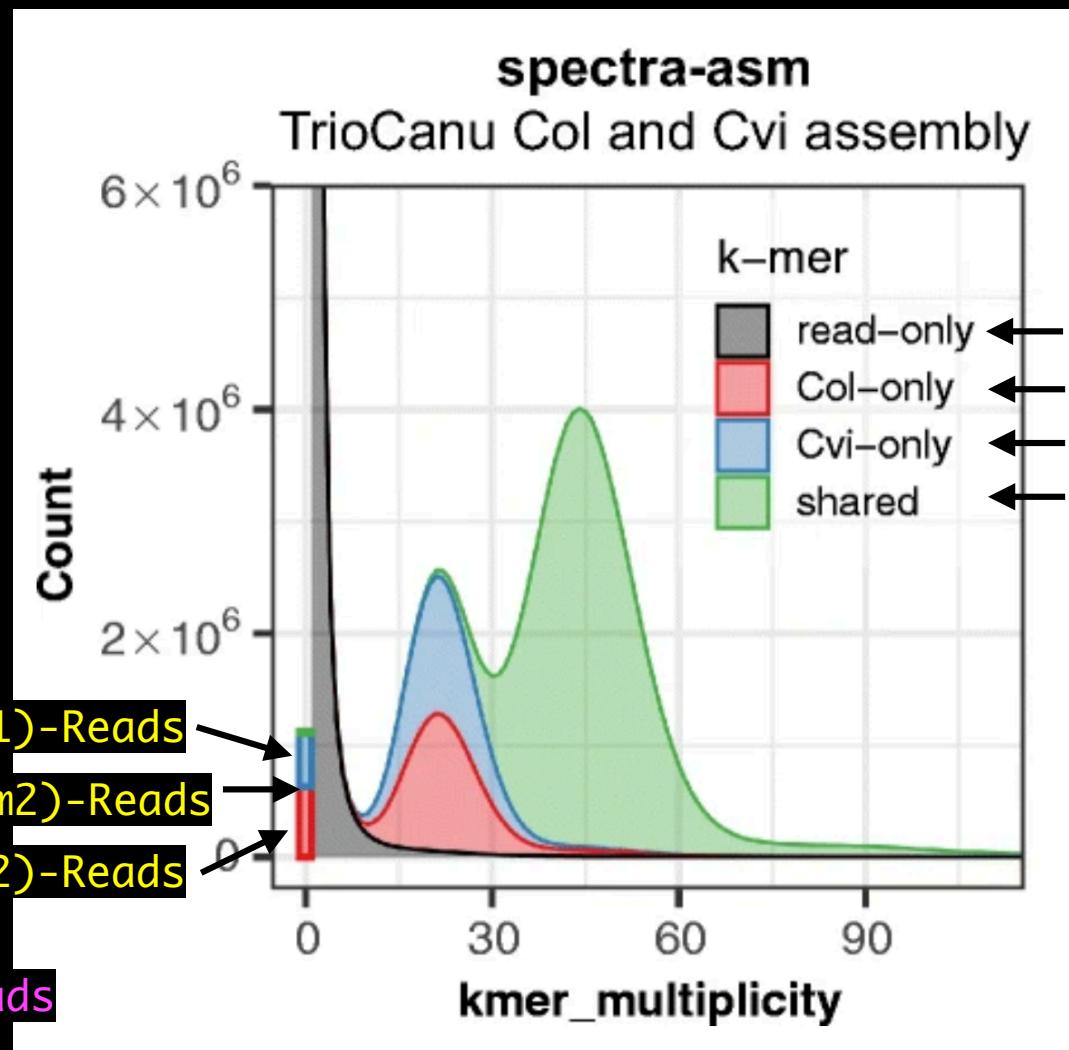
```
CNplot [-w<double(6.0)>] [-h<double(4.5)>] [-z] [-lfs]
[-[xX]<number(x2.1)>] [-[yY]<number(y1.1)>] [-pdf]
<out>[.cni]
```

Produces just plots and requires no significant computation.  
(Often a user wants to just adjust the size or aspect or style of the plot)

# Assembly Spectra Plots:

-k as for CNplot save  
extension is different

```
ASMplot [-w<double(6.0)>] . . . [-P<dir(/tmp)>]  
<reads>[.ktab] <asm1:dna> [<asm2:dna>] <out>[.asmi]
```

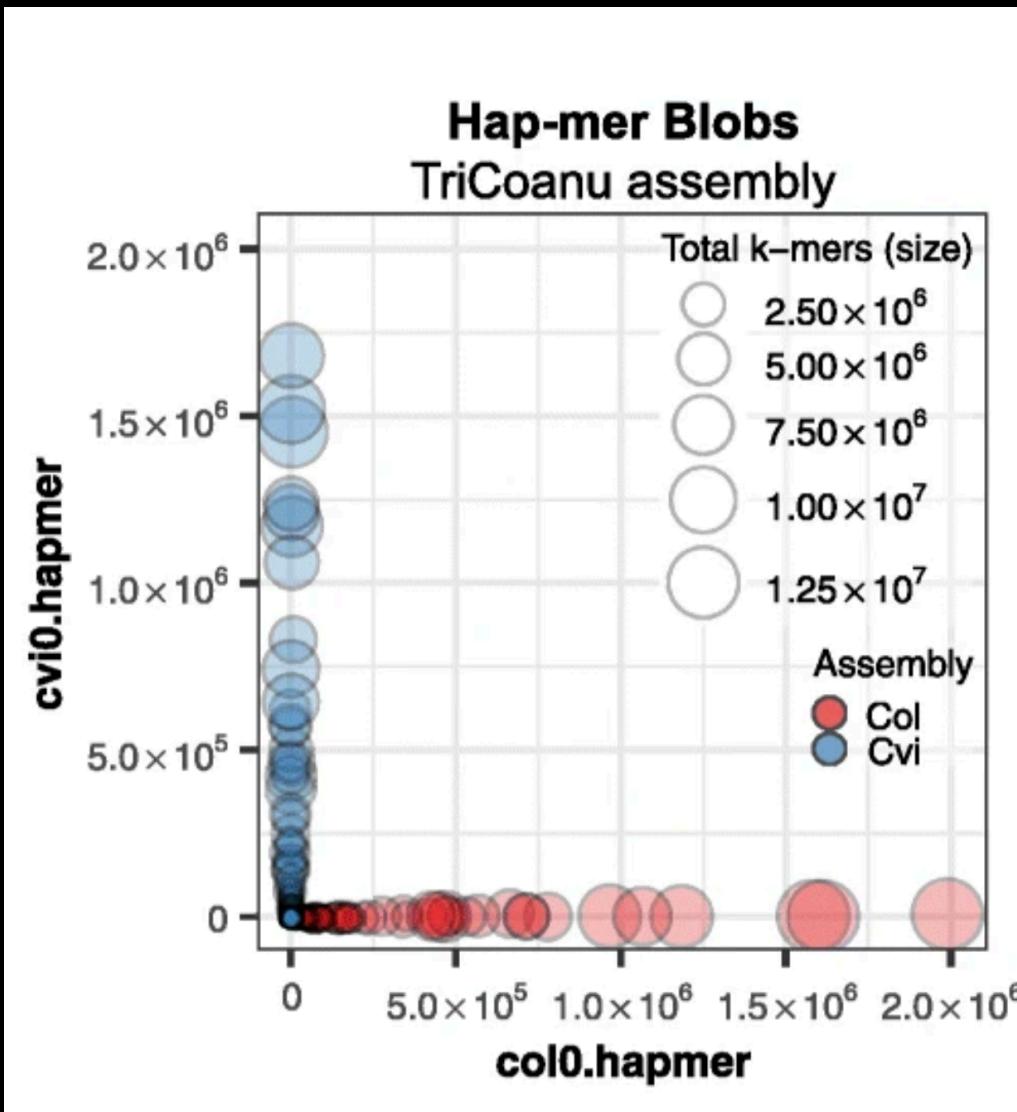


Reads - #(Asm1|Asm2)  
Reads &. (Asm1-Asm2)  
Reads &. (Asm2-Asm1)  
Reads &. #(Asm2&Asm1)

Reads - Asm1  
Reads &. Asm1

# Hap-mer Blob Plot: Are the contigs well phased?

```
HAPplot[-w<double(6.0)>] [-h<double(4.5)>] [-pdf]
[-vk][-T<int(4)>] [-P<dir(/tmp)>]
<mat>[.hap[.ktab]] <pat>[.hap[.ktab]] <asm1:dna> [<asm2:dna>] <out>[.hpi]
```



1 blob per contig whose radius is proportional to its length

$M(c) = \# \text{ of k-mers in mat.hap.ktab}$

$P(c) = \# \text{ of k-mers in mat.hap.ktab}$

position of blob center =  $(M(c), P(c))$

opacity =  $(M(c)+P(c))/2|c|$

color = which assembly

# MercuryFK Outputs

MercuryFK Read Pat Mat Asm1 Asm2 Out

## Asm1 only

- Out.Asm1.spectra-cn.fl.png
- Out.Asm1.spectra-cn.ln.png
- Out.Asm1.spectra-cn.st.png
- Out.Asm1.qv
- Out.Asm1\_only.bed
  
- Out.spectra-asm.fl.png
- Out.spectra-asm.ln.png
- Out.spectra-asm.st.png
- Out.qv
- Out.completeness.stats

## + Asm2

- Out.Asm2.spectra-cn.fl.png
- Out.Asm2.spectra-cn.ln.png
- Out.Asm2.spectra-cn.st.png
- Out.Asm2.qv
- Out.Asm2\_only.bed

- Out.spectra-cn.fl.png
- Out.spectra-cn.ln.png
- Out.spectra-cn.st.png

## Asm1 + Trio

- Out.Asm1.Mat.spectra-cn.fl.png
- Out.Asm1.Mat.spectra-cn.ln.png
- Out.Asm1.Mat.spectra-cn.st.png
- Out.Asm1.Pat.spectra-cn.fl.png
- Out.Asm1.Pat.spectra-cn.ln.png
- Out.Asm1.Pat.spectra-cn.st.png
- Out.Asm1.phased\_block.blob.png
- Out.Asm1.phased\_block.bed
- Out.Asm1.phased\_block.stats
- Out.Asm1.block.N.png
- Out.Asm1.continuity.N.png
  
- Out.hapmers.blob.png

## All

- Out.Asm2.Mat.spectra-cn.fl.png
- Out.Asm2.Mat.spectra-cn.ln.png
- Out.Asm2.Mat.spectra-cn.st.png
- Out.Asm2.Pat.spectra-cn.fl.png
- Out.Asm2.Pat.spectra-cn.ln.png
- Out.Asm2.Pat.spectra-cn.st.png
- Out.Asm2.phased\_block.blob.png
- Out.Asm2.phased\_block.bed
- Out.Asm2.phased\_block.stats
- Out.Asm2.block.N.png
- Out.Asm2.continuity.N.png

CN spectra of <asm> vs <hapmer>

CN spectra of Asm1 U Asm2 (if both)

# MercuryFK QV Metric

$T$  = # of k-mers in Asm

$S$  = # of k-mers in Reads&Asm

$\Pr(\text{k-mer is not an error}) = (1-\varepsilon)^k$

$\Pr(\text{k-mer is not an error}) = S/T \quad (\text{k-mers supported by reads})$

$$\Rightarrow \varepsilon = 1 - (S/T)^{1/k}$$

$$qv = 10 \log_{10} \varepsilon$$

$$\text{Out.Asm.qv} \equiv T - S + T \cdot \varepsilon \cdot qv$$

$\text{Out.Asm\_only.bed} = \text{intervals covered by k-mers in Asm-Reads}$   
 $= \text{places where there might be an error}$

$\text{Out.qv} = \text{cumulative over all assemblies}$

# MercuryFK Outputs

MercuryFK Read Pat Mat Asm1 Asm2 Out

## Asm1 only

- Out.Asm1.spectra-cn.fl.png
- Out.Asm1.spectra-cn.ln.png
- Out.Asm1.spectra-cn.st.png
- Out.Asm1.qv
- Out.Asm1\_only.bed
  
- Out.spectra-asm.fl.png
- Out.spectra-asm.ln.png
- Out.spectra-asm.st.png
- Out.qv
- Out.completeness.stats

## + Asm2

- Out.Asm2.spectra-cn.fl.png
- Out.Asm2.spectra-cn.ln.png
- Out.Asm2.spectra-cn.st.png
- Out.Asm2.qv
- Out.Asm2\_only.bed
  
- Out.spectra-cn.fl.png
- Out.spectra-cn.ln.png
- Out.spectra-cn.st.png

## Asm1 + Trio

- Out.Asm1.Mat.spectra-cn.fl.png
- Out.Asm1.Mat.spectra-cn.ln.png
- Out.Asm1.Mat.spectra-cn.st.png
- Out.Asm1.Pat.spectra-cn.fl.png
- Out.Asm1.Pat.spectra-cn.ln.png
- Out.Asm1.Pat.spectra-cn.st.png
- Out.Asm1.phased\_block.blob.png
- Out.Asm1.phased\_block.bed
- Out.Asm1.phased\_block.stats
- Out.Asm1.block.N.png
- Out.Asm1.continuity.N.png
  
- Out.hapmers.blob.png

## All

- Out.Asm2.Mat.spectra-cn.fl.png
- Out.Asm2.Mat.spectra-cn.ln.png
- Out.Asm2.Mat.spectra-cn.st.png
- Out.Asm2.Pat.spectra-cn.fl.png
- Out.Asm2.Pat.spectra-cn.ln.png
- Out.Asm2.Pat.spectra-cn.st.png
- Out.Asm2.phased\_block.blob.png
- Out.Asm2.phased\_block.bed
- Out.Asm2.phased\_block.stats
- Out.Asm2.block.N.png
- Out.Asm2.continuity.N.png

# MerquryFK Completeness

$T$  = # of k-mers in Rel(Reads)

$S$  = # of k-mers in Rel(Reads)&Asm

$S/T$  is fraction of Asm covered by reliable read k-mers

Out.completeness.stats  $\equiv$  target · source ·  $S \cdot T \cdot 100(S/T)$

target = Asm1, Asm2, ‘both’

source = ‘all’, Mat, Pat

# MerquryFK Phasing

With trio data, can find runs of pat or mat hap-mers in the scaffolds of an assembly = hap-mer blocks (of the same phase)

Small switches of phase are allowed between reliable blocks (= 20Kbp and/or 5 hap-mers in a row).

A well phased assembly has large blocks all of the same kind.

Out.Asm1.phased\_block.stats = #blocks · sum · min · avg · N50 · max

Out.Asm1.phased\_block.bed = block intervals

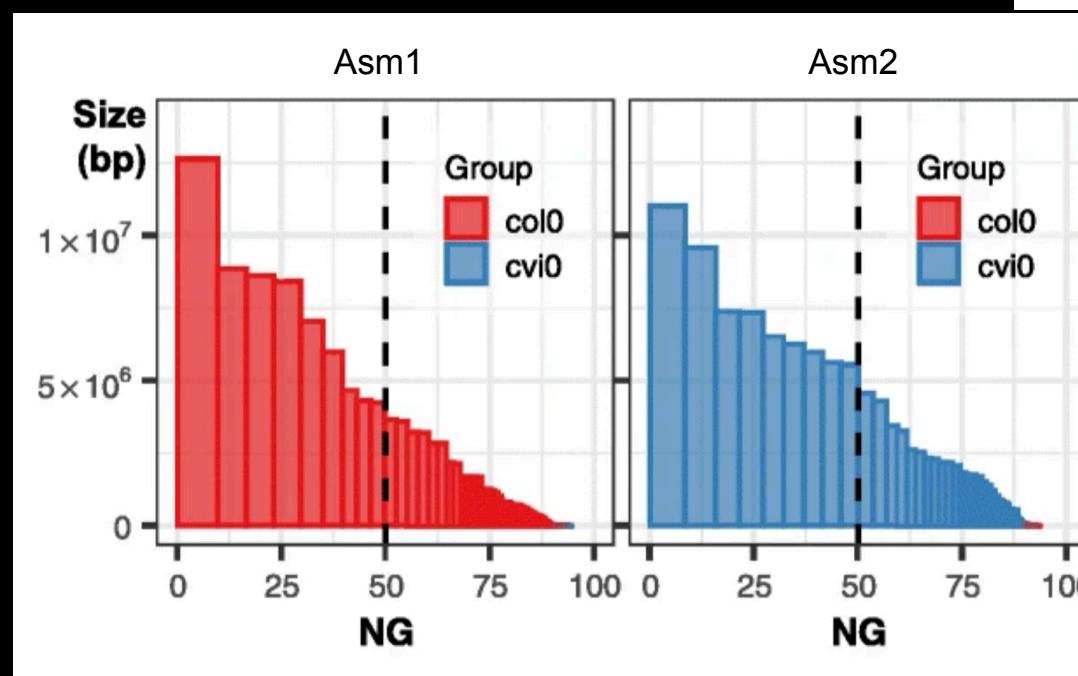
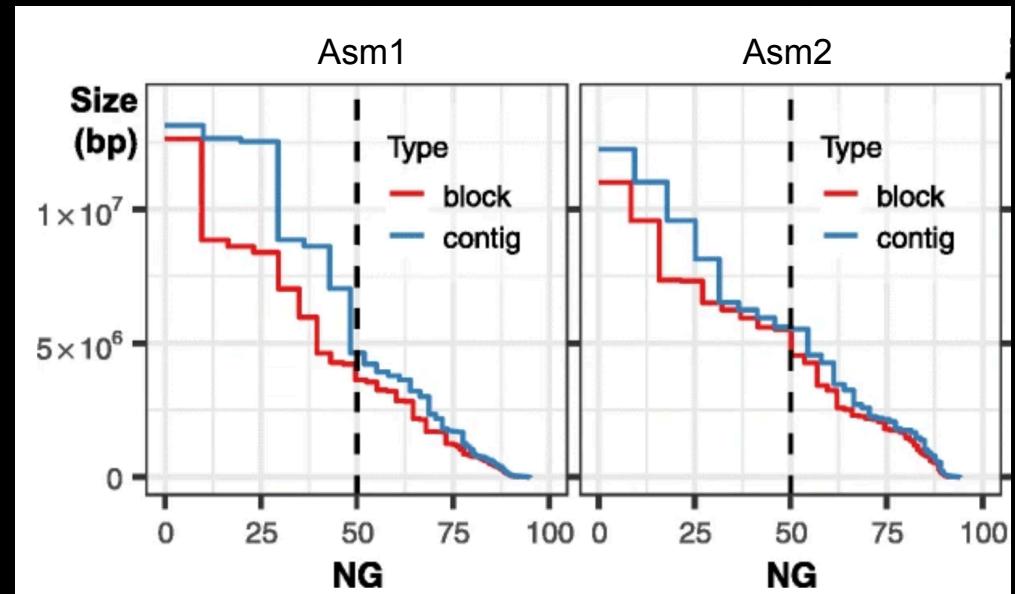
= scaffold · beg · end · phase · purity · switches · #hap-mers

Scaffold	Start	End	Phase	Purity	Switches	Markers
1	0	24973	Gut4	0.463	43	9289
1	26394	30482	Gut3	11.364	25	220
1	30970	846998	Gut4	0.134	449	336080
1	847235	847415	Gut3	0.000	0	112
1	847708	1404608	Gut4	0.080	162	202012
2	1404621	1456858	Gut4	0.125	21	16795
2	1457275	1457620	Gut3	0.000	0	21
3	1458249	1617417	Gut4	0.000	0	53070
4	1618096	2072629	Gut4	0.136	213	156733
4	2075958	2081903	Gut3	0.000	0	47
5	2082003	3500257	Gut4	0.089	400	450404
6	3500357	5864720	Gut4	0.046	367	86355
7	5864820	6131671	Gut4	0.054	49	79745
8	6131771	10025579	Gut4	0.023	262	1133157
9	10026565	15600117	Gut4	0.038	452	1179938
10	15600217	15601057	Gut3	0.000	0	21
10	15601994	15813295	Gut4	0.076	21	27604
10	15813690	15813726	Gut3	0.000	0	22

... .

# MerquryFK: NG plots

The size distribution of every contig, scaffold and phase blocked is captured in a “NG” continuity plot for each assembly



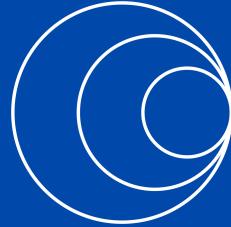
Similarly phase colored hapmer blocks are plotted in order of size.

# MercuryFK: Options

```
MercuryFK [-w<double(6.0)>] [-h<double(4.5)>] [-z] [-lfs]
[-[xX]<number(x2.1)>] [-[yY]<number(y1.1)>] [-pdf]
[-vk][-T<int(4)>] [-P<dir(/tmp)>]
<read>[.ktab] [<mat>[.hap[.ktab]] <pat>[.hap[.ktab]] ]
<asm1:dna> [<asm2:dna>] <out>
```

Same as CNplot! Passed through to FastK, Logex, and internal calls to CNplot, ASMplot, and HAPplot.

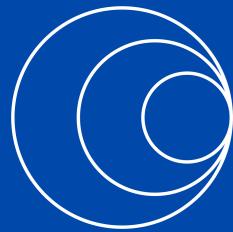
As before the -k option causes all .cni, .asm, and .hpi intermediates to be saved for subsequent replotting.



wellcome  
connecting  
science

# Mercury Exercises

Will run it on a few small data sets  
and look at the results



wellcome  
connecting  
science

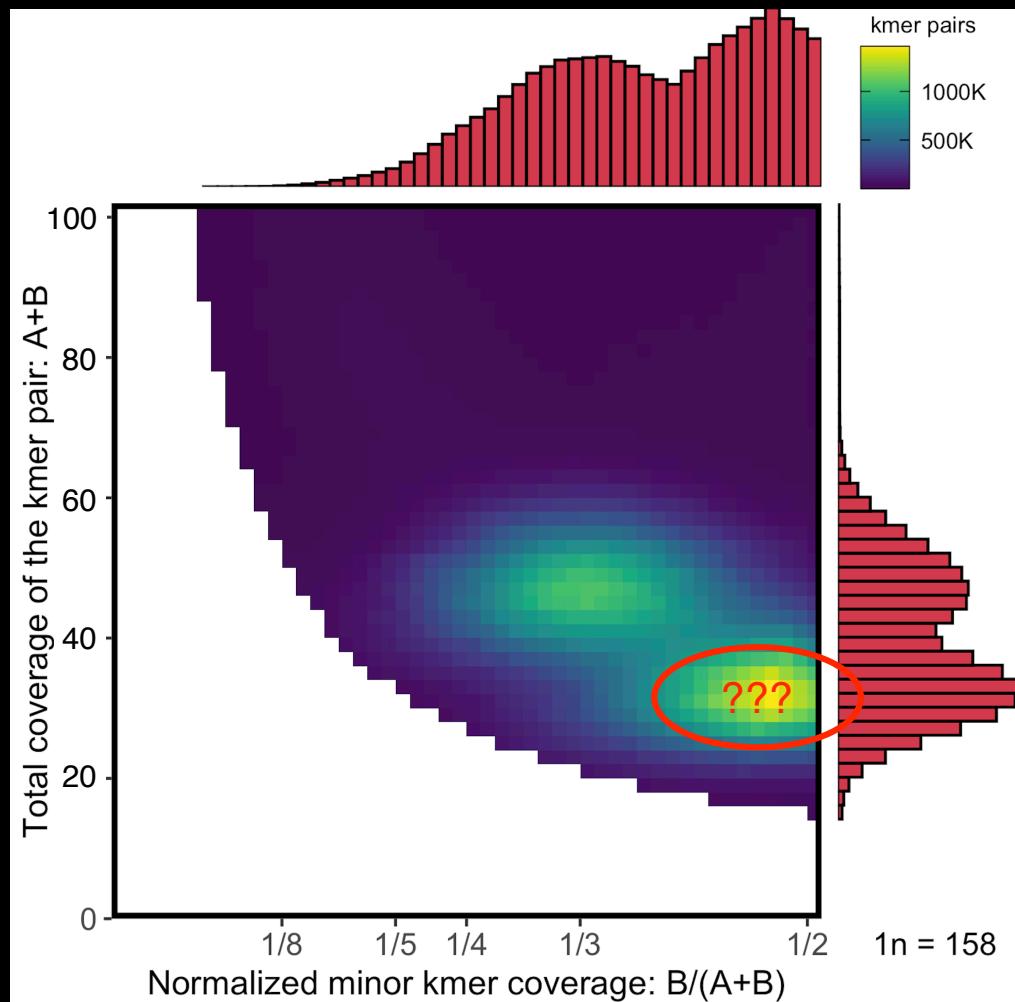
# SmudgePlot 2.0

If time permits, optional.

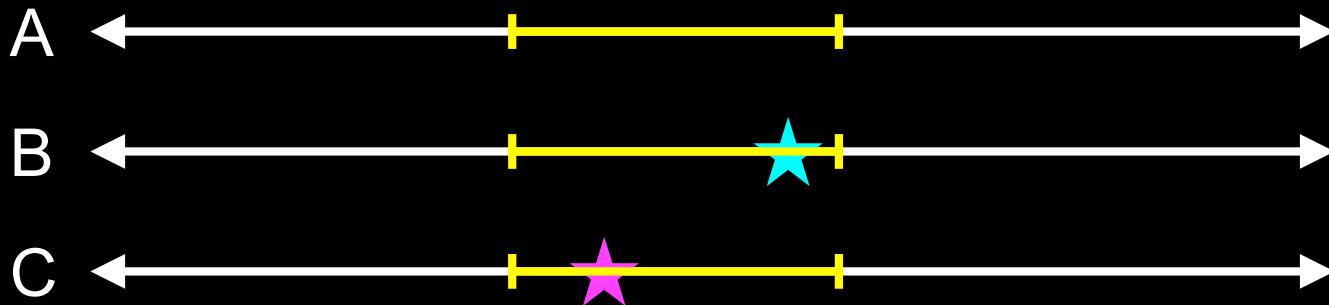
# SmudgePlot ‘C-Backend’

Simulated 50X triploid data set  
with 1% heterogeneity

*Original SmudgePlot*



# The explanation:



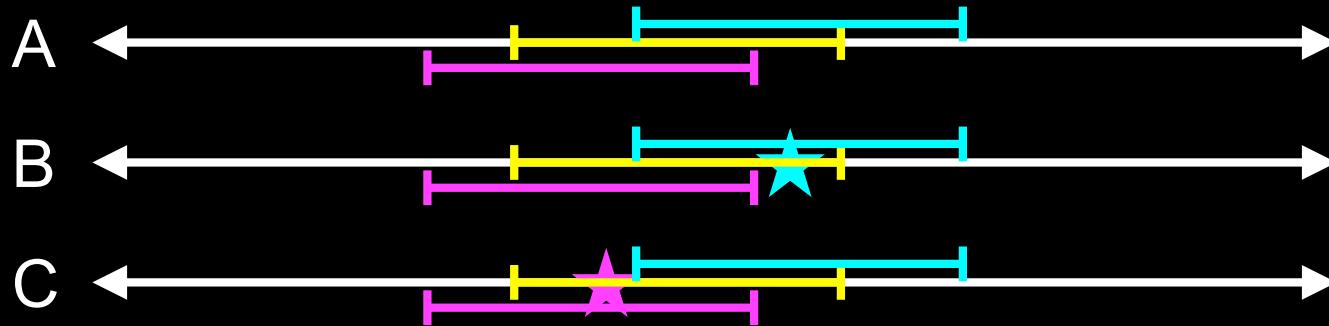
$$\text{cnt}(A) = \text{cnt}(B) = \text{cnt}(C) \sim 1/3c$$

$$A/(A+B) \sim A/(A+C) \sim B/(B+C) \sim 1/2$$

A,B / A,C / B,C contribute to the diploid smudge!

The greater k or the heterogeneity the  
greater the effect

# The fix:

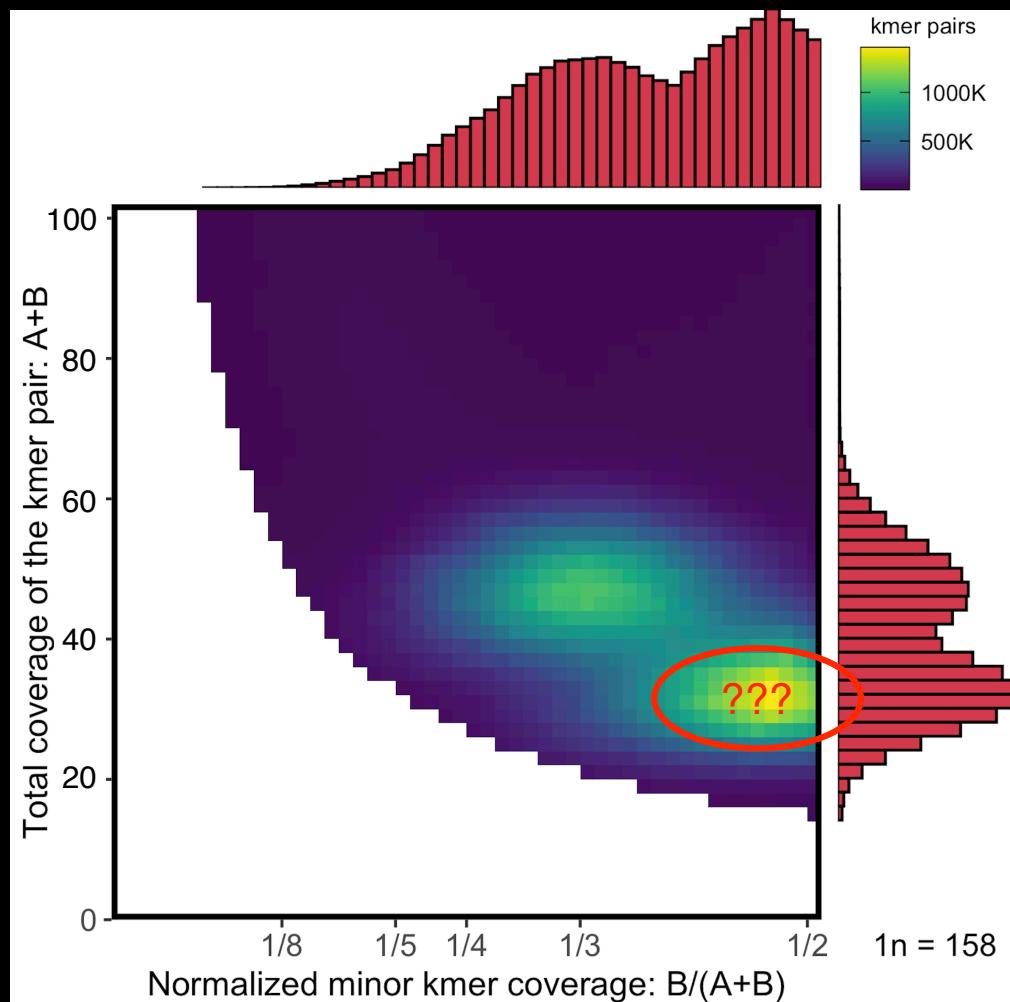


If a k-mer occurs in more than 1 non-erroneous het-pair then don't count it.

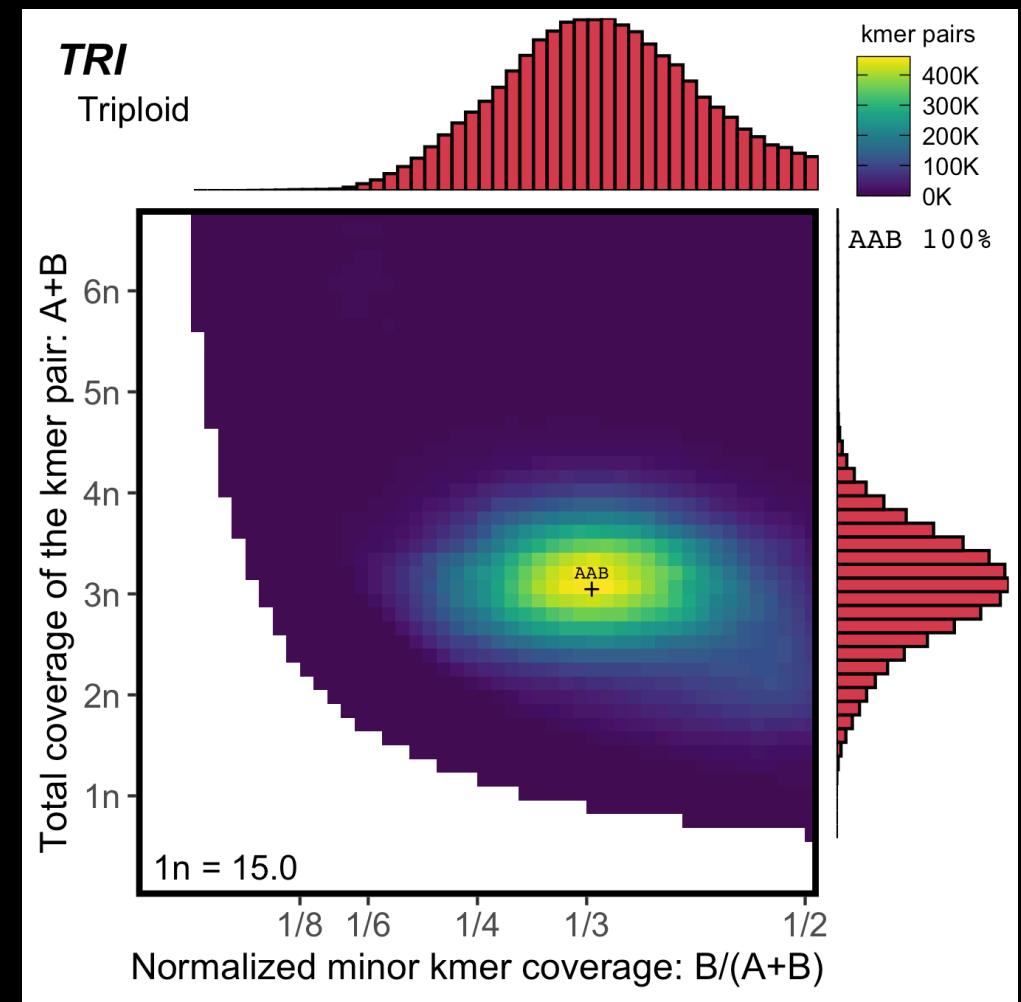
# SmudgePlot 2.0

Simulated 50X triploid data set  
with 1% heterogeneity

*Original SmudgePlot*



*SmudgePlot 2.0*



```
smudgeplot.py hetmers -L # <source:.ktab>
```

All k-mers with count less than -L are considered error

Table needs to be trimmed (to -L) and symmetric before finding pairs

User can do it, or smudgeplot will – determined automatically

Assuming **foo** is an untrimmed, canonical table:

```
smudgeplot.py -t 8 -L 4 foo.ktab    331s
```

```
Logex -T8 'trim=A[4-]' foo          97s      328s  
smudgeplot.py -t 8 -L 4 trim.ktab   231s
```

```
Logex -T8 'trim=A[4-]' foo          97s      335s  
Symmex -T8 trim symx               56s  
smudgeplot.py -t 8 -L 4 symx.ktab   182s
```

Always trim first!

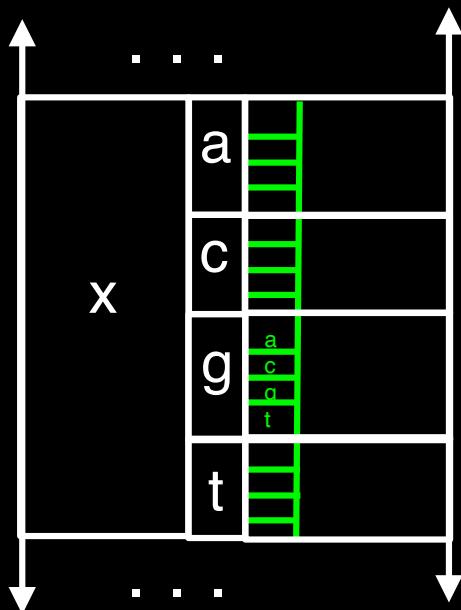
```
Symmex -T8 foo symx                640s      works but slow  
smudgeplot.py -t 8 -L 4 symx.ktab   482s
```

# Smudgeplot Algorithm:

Find all k-mer pairs  $xay, xby$  (w. counts A,B): plot  $\min(A,B)/A+B$

In a *symmetric* table of k-mers:

For all  $x$ ,  $|x| < k$ :



## Threading:

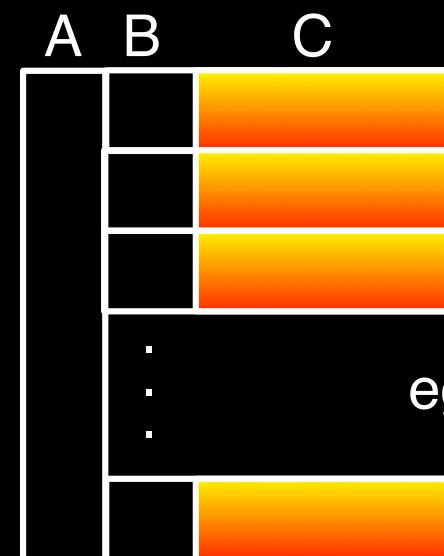
- A. If  $|lx|$  small: divide lists into N-parts
  - B. If  $|lx|$  not small: one thread, recurse
  - C. If  $|lx|$  large: table range is in core

$O(kN/T)$  time

**O( $kN$ ) time**

# Merge 4 lists looking for = suffixes

Also find subdivisions  
for next lower level



eg: A 1-3  
B 4-5  
C 6-..