Alignment-free detection of copy number variations (CNVs) using strongly unique *k*-mers and fused lasso regularization

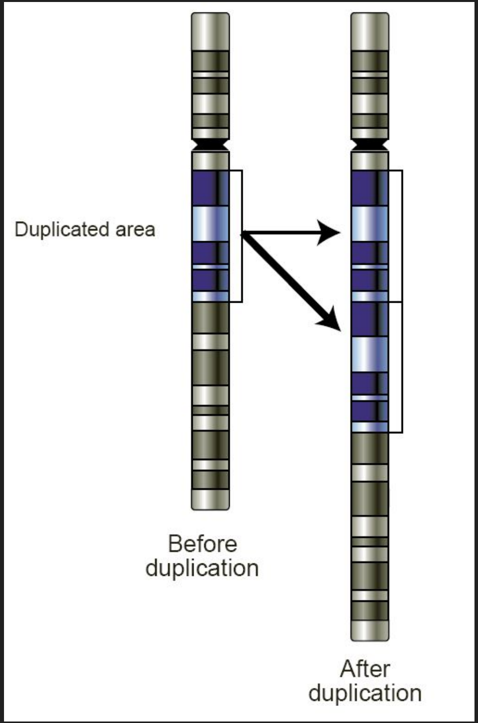
Till Hartmann[1,2], Elias Kuthe[2,3], Alicia Tüns[2,4], Alexander Schramm[2,4], Jens Zentgraf[2,3], Sven Rahmann[1,2,3]

(DSB online, 12-Feb-2021)

[1] Genome Informatics, Institute of Human Genetics, University of Duisburg-Essen, Essen, Germany
[2] Collaborative Research Center SFB 876, Dortmund/Essen, Germany
[3] Bioinformatics, Computer Science XI, TU Dortmund University, Dortmund, Germany
[4] Laboratory for Molecular Oncology, Department of Medical Oncology, West German Cancer Center, University Hospital Essen, University of Duisburg-Essen, Essen, Germany

copy number variations (CNVs)

- segmental duplications & amplifications
- segmental deletions & losses
- may change gene copy numbers
- may influence gene expression
- frequently happen in cancer cells

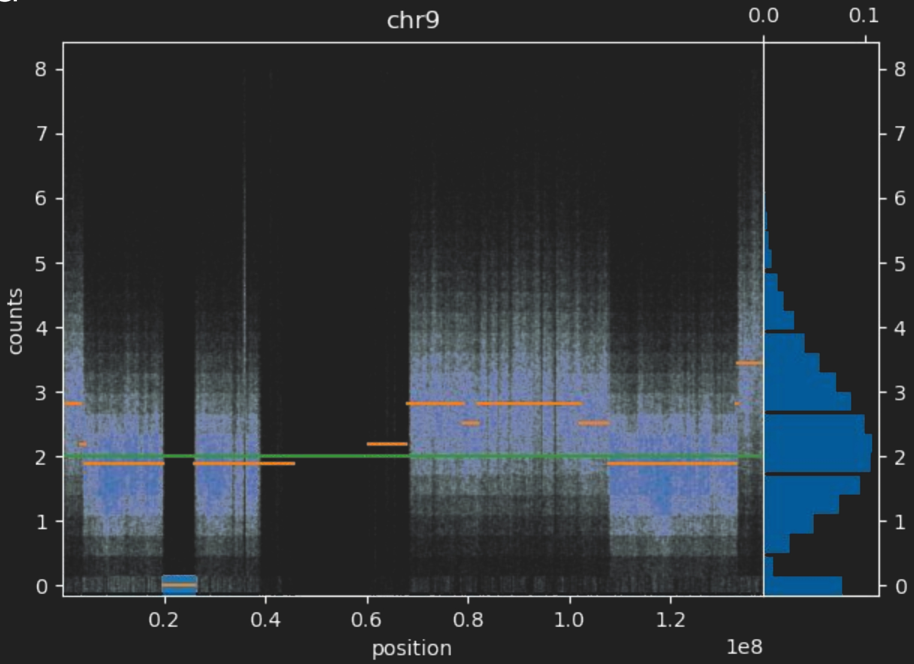


source: genome.gov

why alignment-free CNV calling?

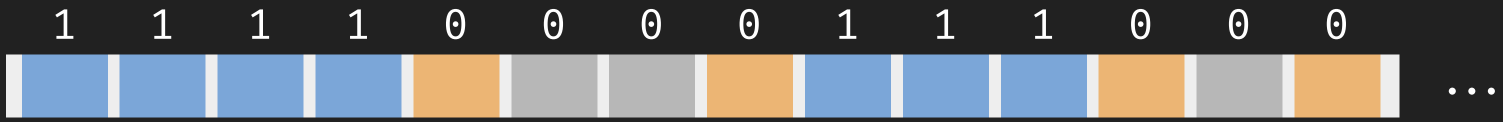
- alignment-free approaches more efficient than mapping-based
- avoids mapping bias
- *k*-mer count: direct CN estimate if *k*-mer is unique in genome

```
GGCTCAGAACCCTGAATTCTAGTCTC
GCGCCCGGCCCTGGGTGGGGAGATAT
AGGTTAGAGATACTCAAGCTCCCCTT
TGTCCCTTTTCTGCGCCTCAAGGGG
TGTGACATGAACAAAACCAAAACCTT
...
```



overview

- determine robust *k*-mer probes:
  strongly *unique* *k-mer* : unique in genome & no hamming distance 1 neighbours
- bit vector indicator of these probes in reference (aka *"index"*)

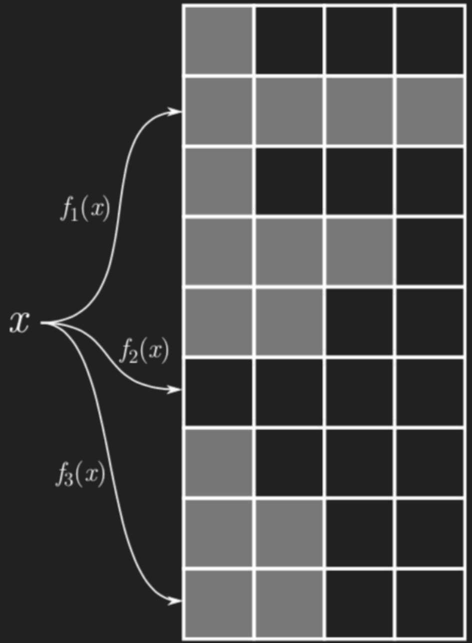


- for each sample:
  - deduplicate raw reads (bloom filter)
  - count *k*-mers (any k-mer counter will do)
  - counts → copy number via k-mer count histogram
  - "measure" copy number signal on robust probes across reference
  - segment signal with fused lasso

Part I:
choosing strongly unique *k*-mer probes

hashing genomic *k*-mers

- Hash and count genomic *k*-mers (up to 2)
- Use 3-way bucketed Cuckoo hashing *
- 3 hash functions, each maps a *k*-mer to a bucket
- each bucket can store up to 4 elements
  (such that a bucket fits within a cache line)
  → 12 possible locations for each element
- at worst 3 memory lookups (cache misses),
  often only 1 or 2, depends on load factor
- use quotienting* for saving space
  (only part of the key needs to be stored)



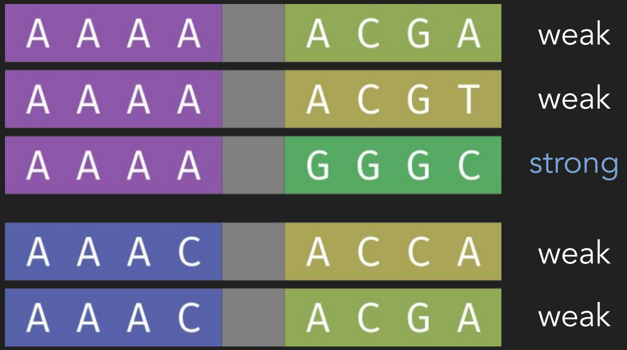
* as featured in: "Fast lightweight accurate xenograft sorting", Jens Zentgraf & Sven Rahmann, WABI 2020
(with an extended version being under review at Algorithms in Molecular Biology)

marking weak vs. strong *k*-mers

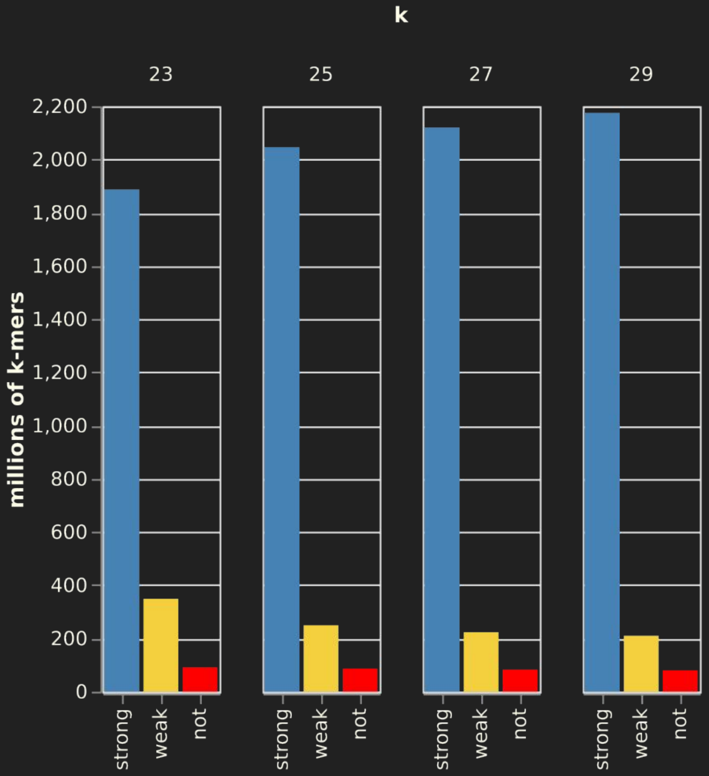
- determine robust probes:
  strongly unique k-mer : unique & has no hamming distance 1 neighbours

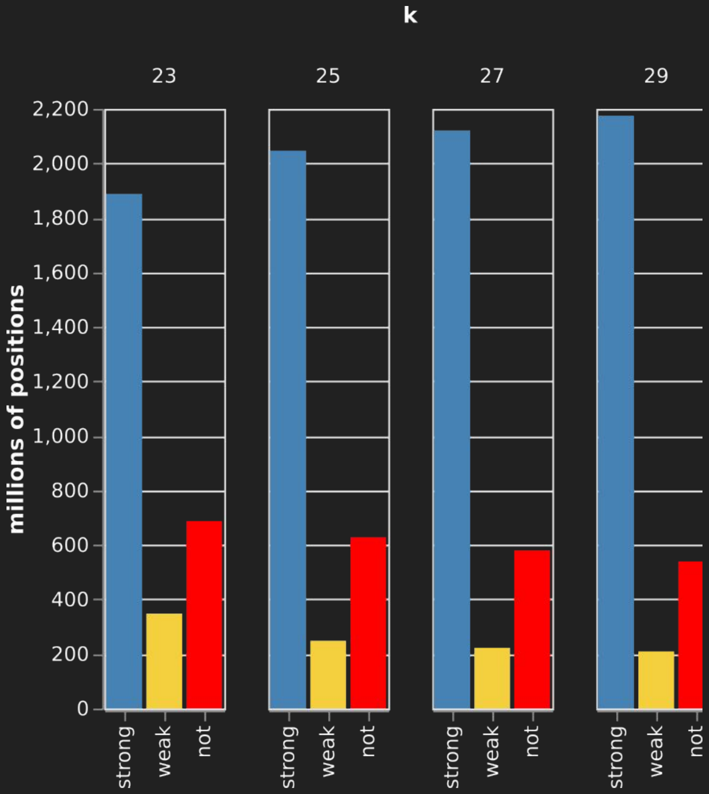
- naïve: for each *k*-mer, look up each of its 3·*k* neighbours,
  mark those with any neighbour as *weak* (remainder as *strong*)

- better:
  - sort list of *k*-mers *and* their reverse complements
  - partition into blocks by *k*-mer prefix of length *k*/2
  - for each block: test all suffix pairs for HD 1.
  - using fast bit-magic test for HD 1
  - blocks can be processed in parallel

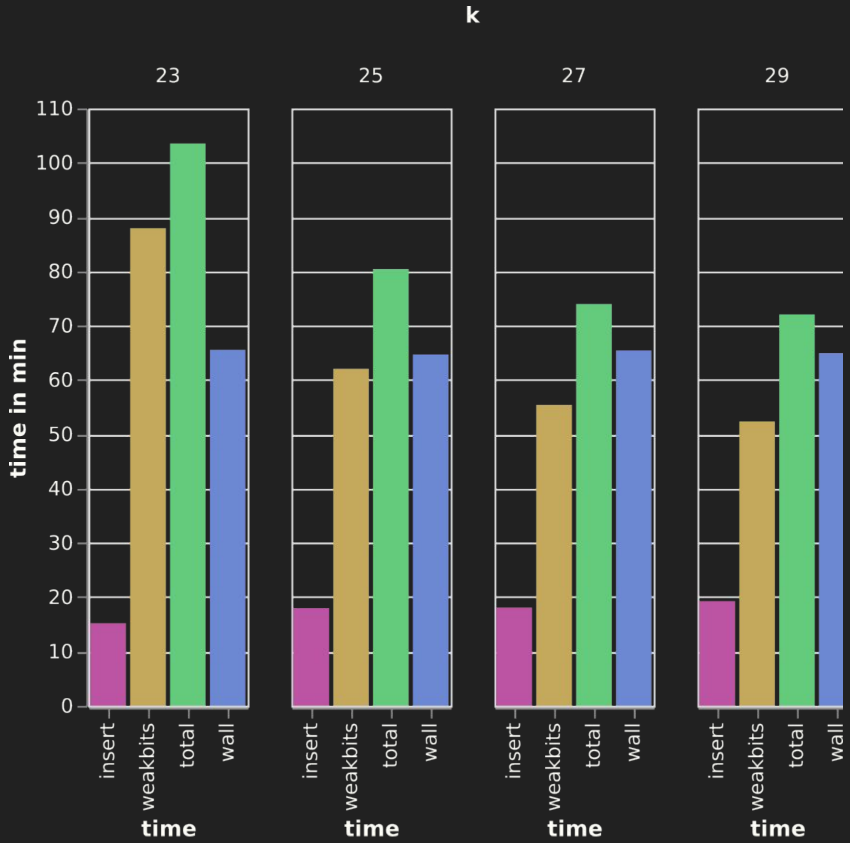


k-mer type distribution → pick *k*=27





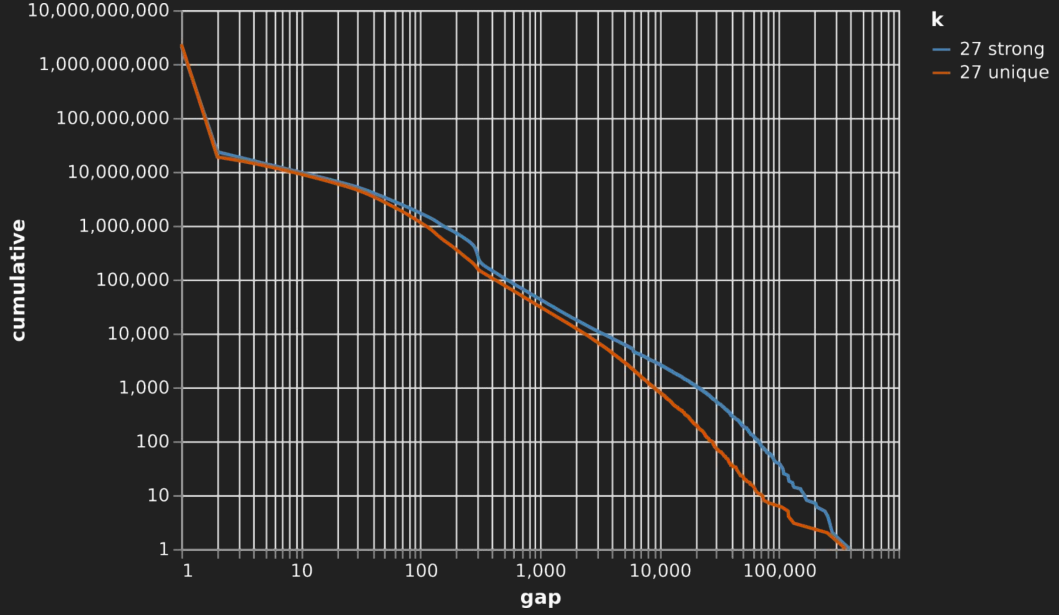
build times (reference *k*-mer hash)



- times in CPU minutes (except wall: wall minutes)

- insert time (serial) increases with number of *k*-mers: 2.33 G (*k*=23) to 2.47 G (*k*=29)
- marking time of weak *k*-mers is dominant, but decreases with *k* (smaller blocks)
- parallelization over blocks more effective for smaller *k* (fewer, but larger blocks)
- wall clock time constant ~65 min
- genome: UCSC "analysis set"

gaps between strong vs. all unique *k*-mers



- cumulative gap length distribution for *k* = 27 (both axes logarithmic)
- not counting invalid *k*-mers (containing Ns)
- very often *k*-mers directly adjacent (gap 1)
- only few long gaps ≥ 10k:
  - strong 27-mers: 2665 gaps
  - all unique 27-mers: 814 gaps
- maximal gap lengths:
  - strong 27-mers: 389,890
  - all unique 27-mers: 362,527
- caused by exact repeats, also affects alignments !

Part II:
calling CNs in WGS samples using strongly unique *k*-mers

sample processing: counting *k*-mers

- deduplicate paired end reads (large bloom filter with very low fpr)
- use *k*-mer counter of choice (kmc3)

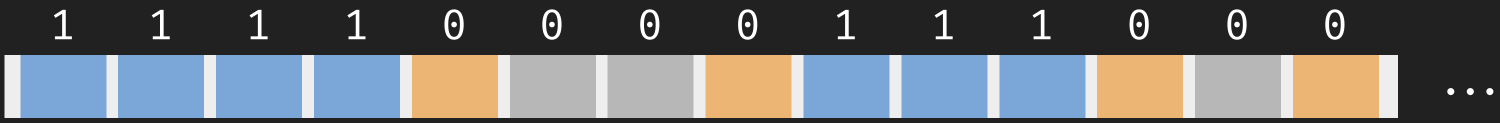
```
GGCTCAGAACCCTGAATTCTAGTCTC
GCGCCCGGCCCTGGGTGGGGAGATAT
AGGTTAGAGATACTCAAGCTCCCCTT
TGTCCCTTTTCTGCGCCTCAAAGGGG
TGTGACATGAACAAAACCAAAACCTT
…
```

$\longrightarrow$

kmer	count
AAAAC	1
AACTT	1
ACGAC	3
ACGAG	1
...	

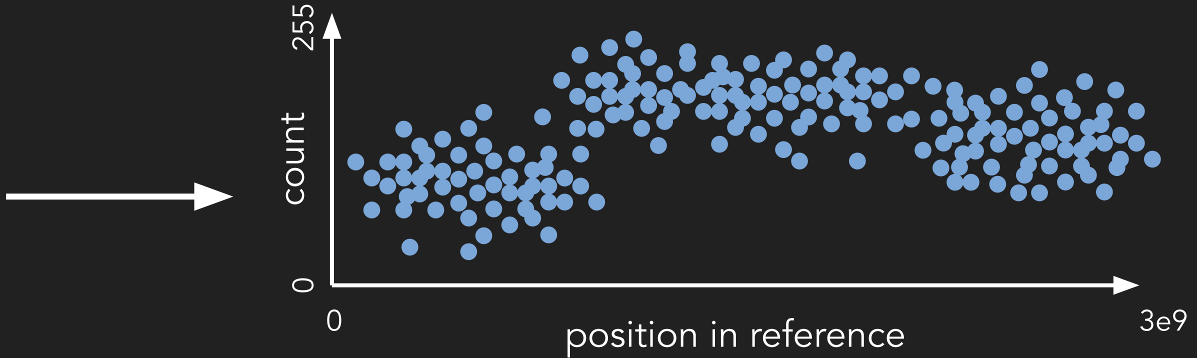
k-mer counts at strong positions

- bitvector of strong k-mers in reference



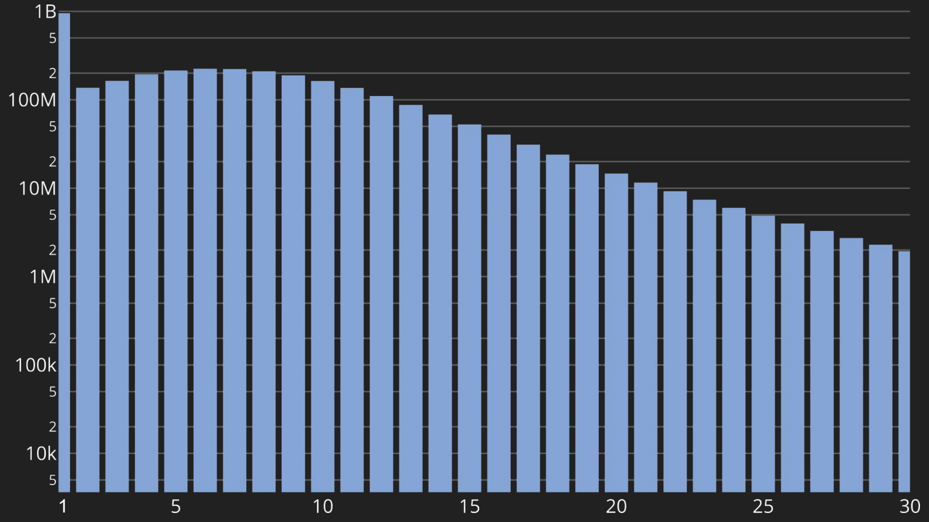
- iterate over reference sequence and bitvector
- for each strongly unique kmer: query sample *k*-mer count (kmc3 API)

kmer	count
AAAAC	1
AACTT	1
ACGAC	3
ACGAG	1
...	



from counts to copy numbers

- find "normcount" corresponding to CN 2 (for diploid genomes) from *k*-mer (log-)count histogram: then CN := 2 · count / normcount
    - fit quadratic polynomial locally around mode (low coverage)
    - fit negative binomial mixture model using EM (high coverage)



segmentation

fused lasso signal approximator (FLSA)

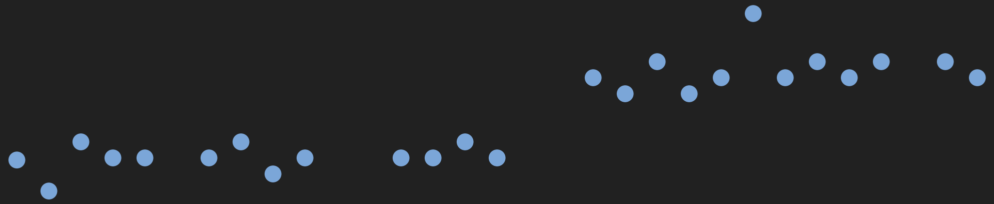
$$f(x) := \sum_{i=1}^{n} \mu_i(\underline{y_i} - \underline{x_i})^2 + \sum_{i=1}^{n-1} \lambda_i \left| x_{i+1} - x_i \right|$$

where

$\lambda_i = \lambda^{(0)} / \mathrm{dist}_i$: weighting factors

dist_i : distance between (strongly unique) *k*-mer *i* and *i* +1 in the reference

$\lambda^{(0)}$: constant, iteratively adapted



segmentation

fused lasso signal approximator (FLSA)

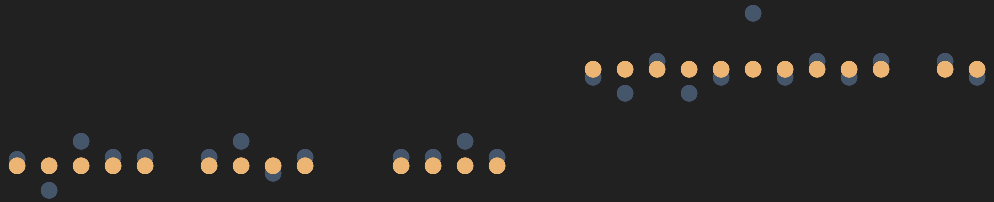
$$f(x) := \sum_{i=1}^{n} \mu_i (\underline{y_i} - \underline{x_i})^2 + \sum_{i=1}^{n-1} \lambda_i \left| x_{i+1} - x_i \right|$$

where

$\lambda_i = \lambda^{(0)} / \mathrm{dist}_i$: weighting factors

dist_i : distance between (strongly unique) *k*-mer *i* and *i* +1 in the reference

$\lambda^{(0)}$: constant, iteratively adapted



segmentation

fused lasso signal approximator (FLSA)

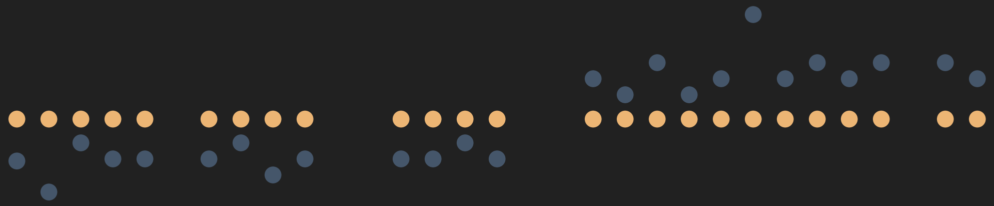
$$f(x) := \sum_{i=1}^{n} \mu_i (\underline{y_i} - \underline{x_i})^2 + \sum_{i=1}^{n-1} \lambda_i \left| x_{i+1} - x_i \right|$$

where

$\lambda_i = \lambda^{(0)} / \mathrm{dist}_i$: weighting factors

dist_i : distance between (strongly unique) *k*-mer *i* and *i* +1 in the reference

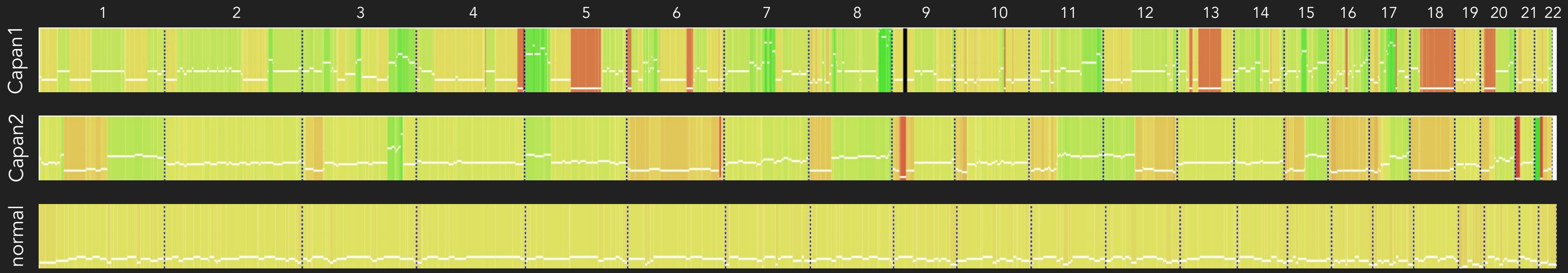
$\lambda^{(0)}$: constant, iteratively adapted



exemplary results

- 2 ~5x coverage WGS samples:
  - Capan1 (Human Pancreatic Adenocarcinoma Cell Line (ATCC HTB-79))
  - Capan2 (Human Pancreatic Adenocarcinoma Cell Line (ATCC HTB-80))
- 1 ~35x coverage WGS sample (normal)

whole genome



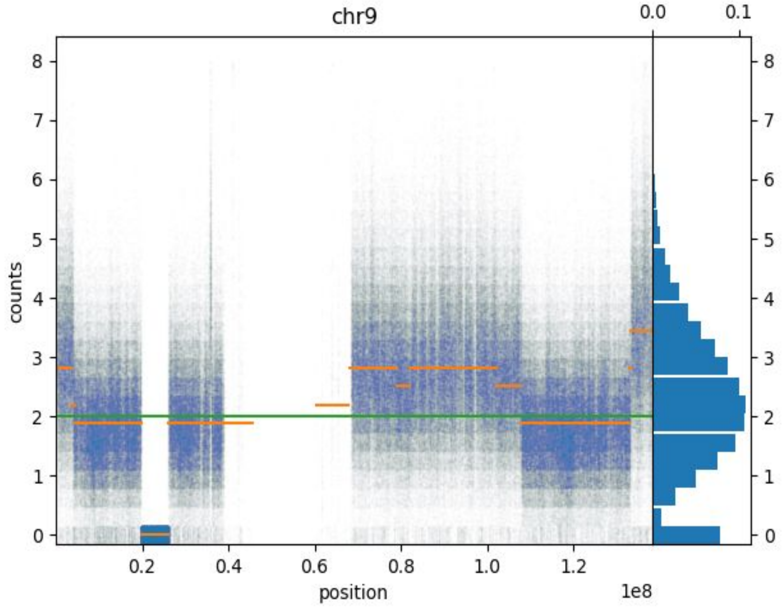
amplification

normal

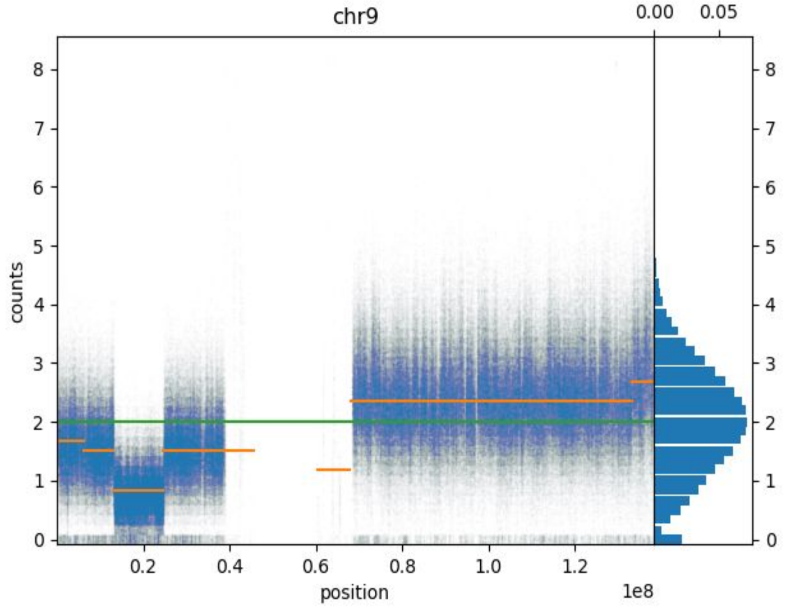
reduction

loss

chromosome details

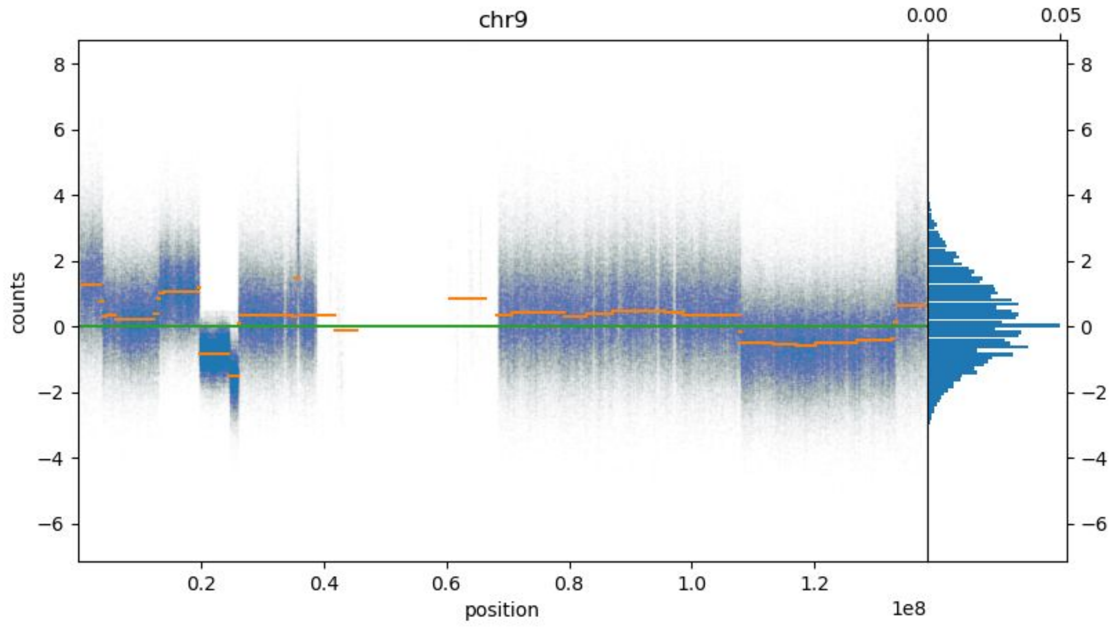


Capan1



Capan2

chromosome details - difference



summary

- what: from raw WGS reads to CNV calls
- how:
  - count strongly unique *k*-mers in sample → copy number signal along genome
  - fused lasso for segmentation of signal
- why:
  - avoid read mapping & alignment:
    save resources (**energy**, cpu hours, memory usage, storage space)
- work in progress:
  - include *k*-mers from known frequent variants
  - further speed-up of weak *k*-mer detection
  - count strongly unique *k*-mers only
  - optimize fused lasso parameters and evaluate on large datasets

appendix
(technical details)

reference genome

- We use the non-redundant "analysis set" from https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/analysisSet/hg38.analysisSet.fa.gz
- note: there may be a problem with the X and Y chromosome, which share some very homologous regions that may lead to non-unique (repeated) k-mers even though we may not want to analyze Y at all.
- note: they hard-masked (Ns) the so-called PAR regions in the analysis set.

memory for reference index

- *k*-mer hashes need between
  7.2 GB for 2.33 G *k*-mers (*k*=23) and 11.3 GB for 2.47 G *k*-mers (*k*=29)
  (3.090 bytes / 23-mer to 4.575 bytes / 29-mer)
  (exact representation, no probabilistic filter, efficient because of quotienting)
- for CNV calling, we only need the bit vector (1 bit / reference position)
  of size approx. 350 MB, independent of *k*
- bit vector may even be compressed (allowing fast left-to-right access)
- but we also need a huge *k*-mer counter of the sample…
  (size depends on sequencing depth, easily over 16 GB)