redkmer: an assembly-free pipeline for the identification of abundant and specific X-chromosome target sequences for X-shredding by CRISPR endonucleases

Philippos Aris Papathanos[1,3], Nikolai Windbichler[2]

September 12, 2017

1. Department of Experimental Medicine, University of Perugia
2. Department of Life Sciences, Imperial College London
3. Corresponding author (p.papathanos@gmail.com)

Supplementary data

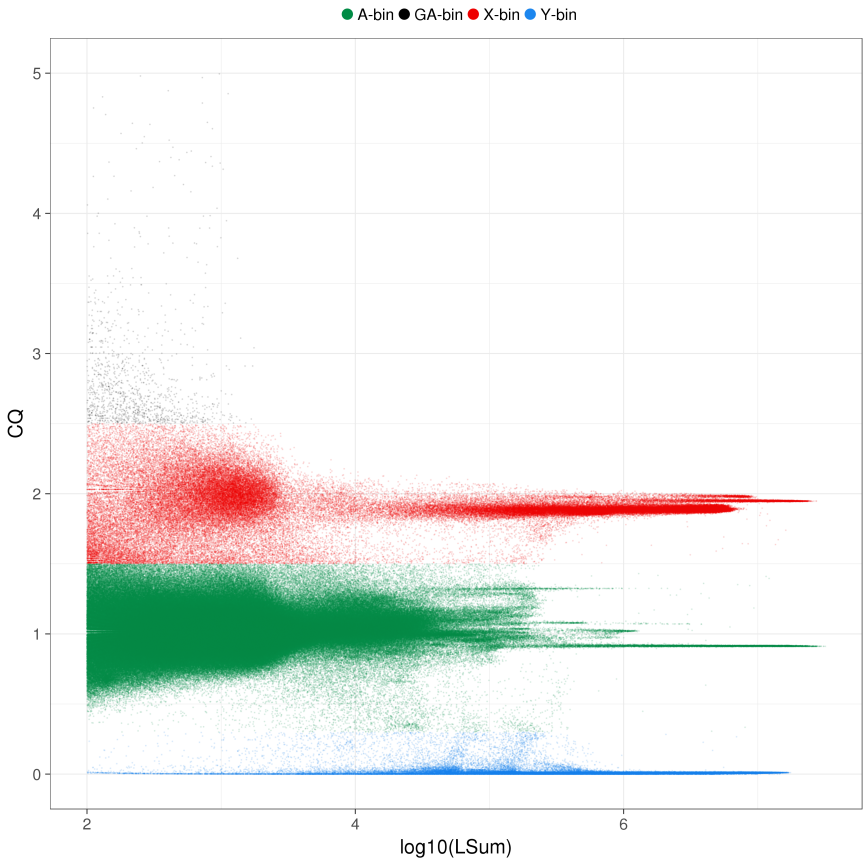
Supplementary Tables

Supplementary Table 1: Important Redkmer input and output files.

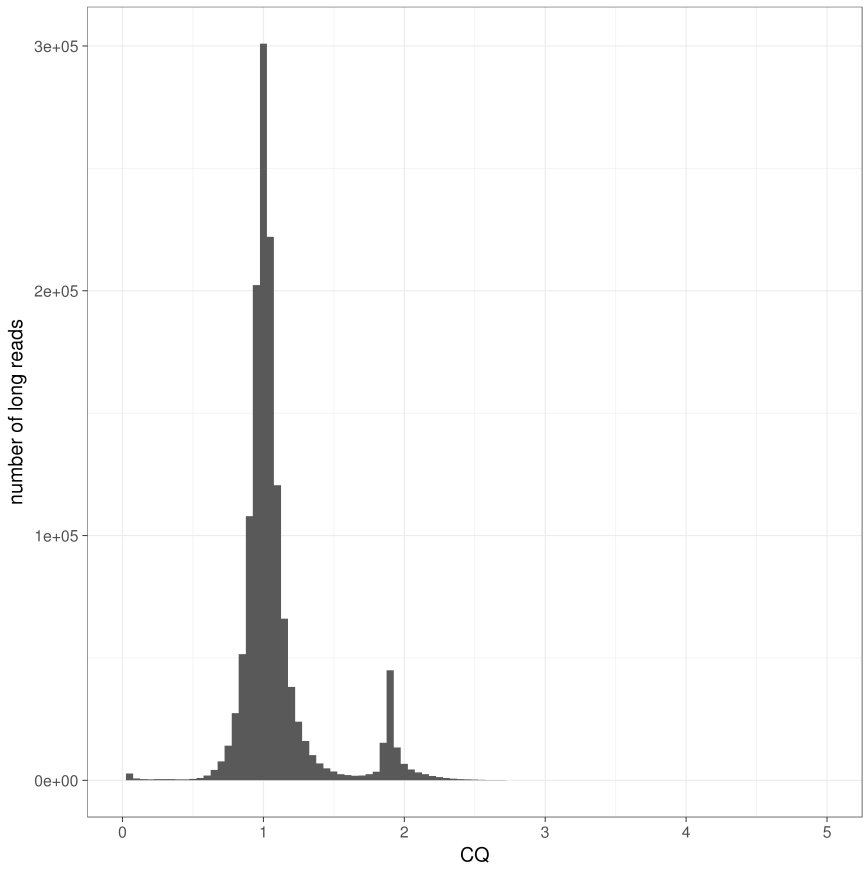
File name(s)	Location	Module	Description
raw_pac.fasta	[df][a]/reads/pacbio	Input	Long reads
raw_m.fastq	[df]/reads/illumina	Input	Male short reads
raw_f.fastq	[df]/reads/illumina	Input	Female short reads
M.fasta	[wd][b]/refgenome/M.fasta	Input	Mitochondrial genome
m_pac.fasta	[df]/reads/pacbio	1	Size filtered long reads
m.fastq	[df]/reads/illumina	1	Mitochondrion-depleted male short reads
f.fastq	[df]/reads/illumina	1	Mitochondrion-depleted female short reads
pacBio_MappedReads.txt	[wd]/pacBio_illmapping	2-3	PacBio mapping data
Xbin; Abin; Ybin; Gabin [c]	[wd]/pacBio_bins/fasta	2-3	Chromosomal bins
allkmers.fasta	[wd]/kmers/fasta	4-5	Kmer sequences + IDs
kmer_results.txt	[wd]/kmers	6-10	Complete kmer results
candidateXkmers.txt	[wd]/kmers	10	X-kmer results
candidateXkmers.fasta	[wd]/kmers	10	X-Kmer sequences
diagnostic plots	[wd]/plots	plotting	Plot output of redkmer

[a] df=data folder
[b] wd=work directory
[c] these are .fasta files

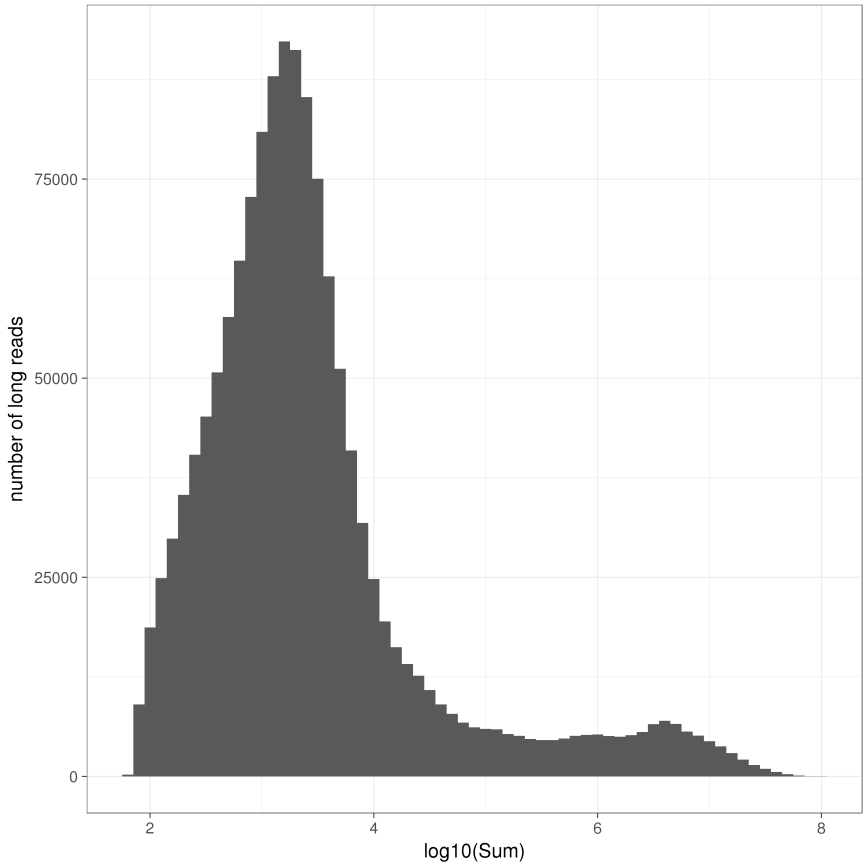
Supplementary Figures



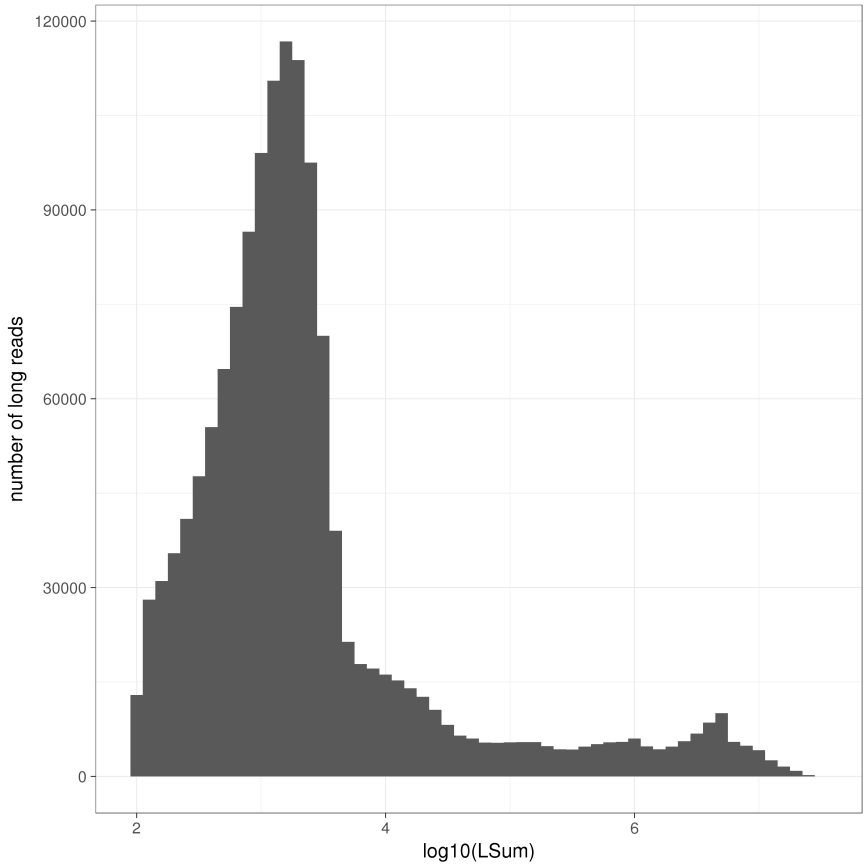
Supplementary Figure 1 - redkmer_plot_reads_1: PacBio read CQ (ratio of female/male data) over log10(LSum) (total number of mapping reads from male and female illumina data) showing chromosomal bins and chromosomal repetitiveness.



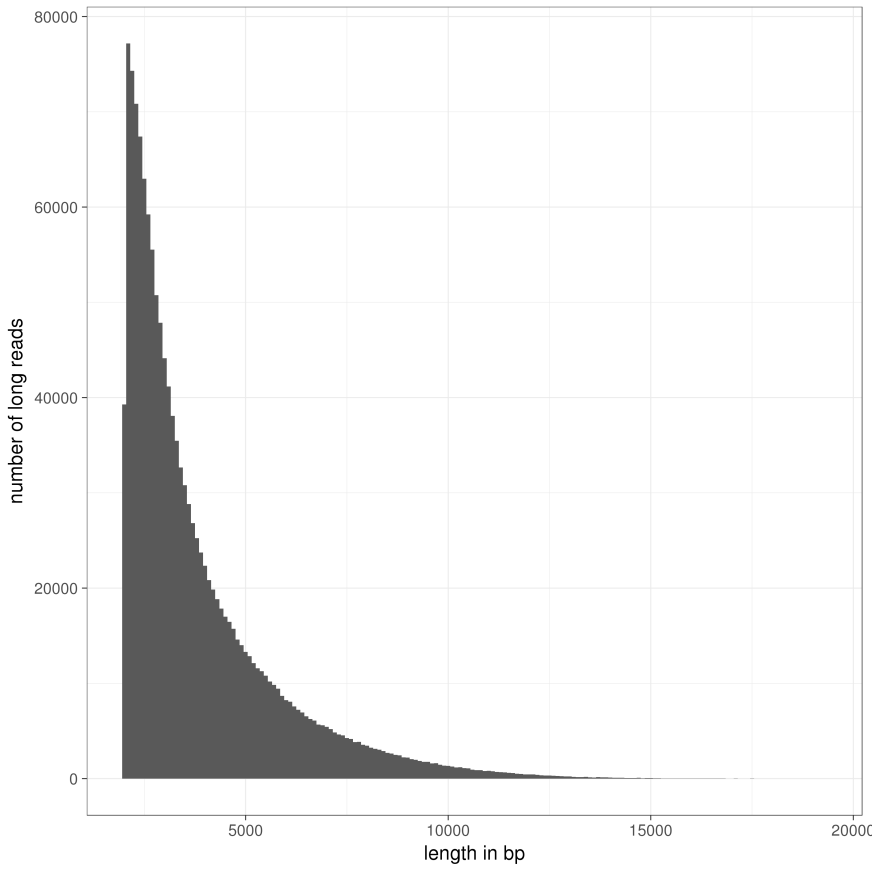
Supplementary Figure 2 - redkmer_plot_reads_2: Histogram of PacBio read CQ distribution. Three distinct peaks can be observed, the major one falling in CQ∼1 (autosome-derived reads), one at CQ∼2 (X-chromosome-derived reads) and one at CQ∼0 (Y-chromosome-derived reads).



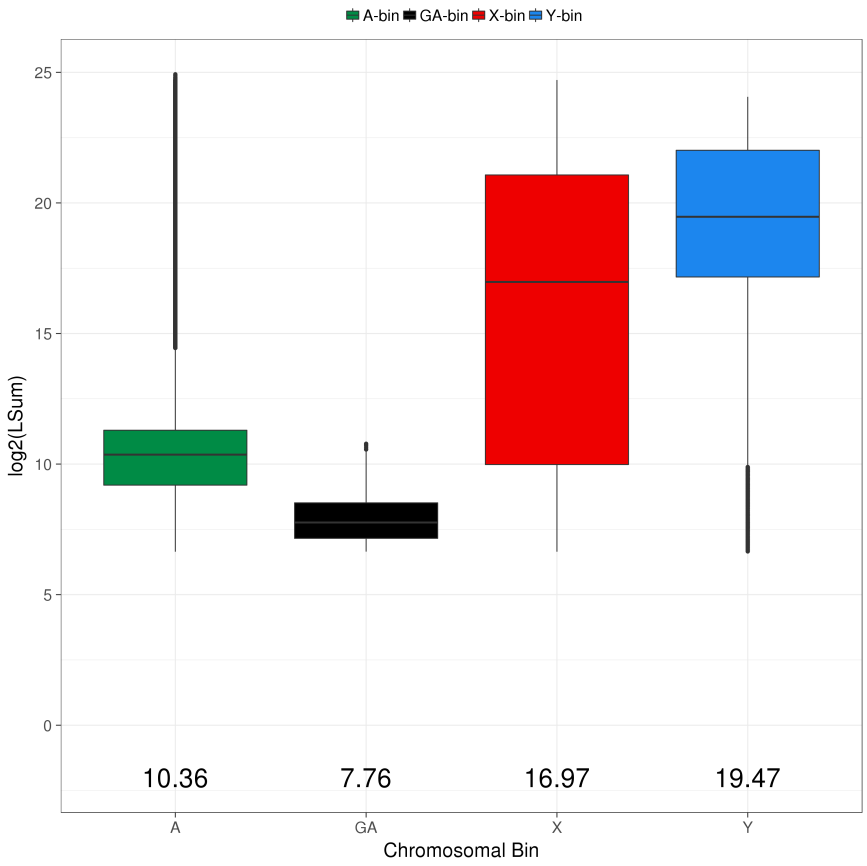
Supplementary Figure 4 - redkmer_plot_reads_3: Histogram of PacBio read log10(Sum) distribution.



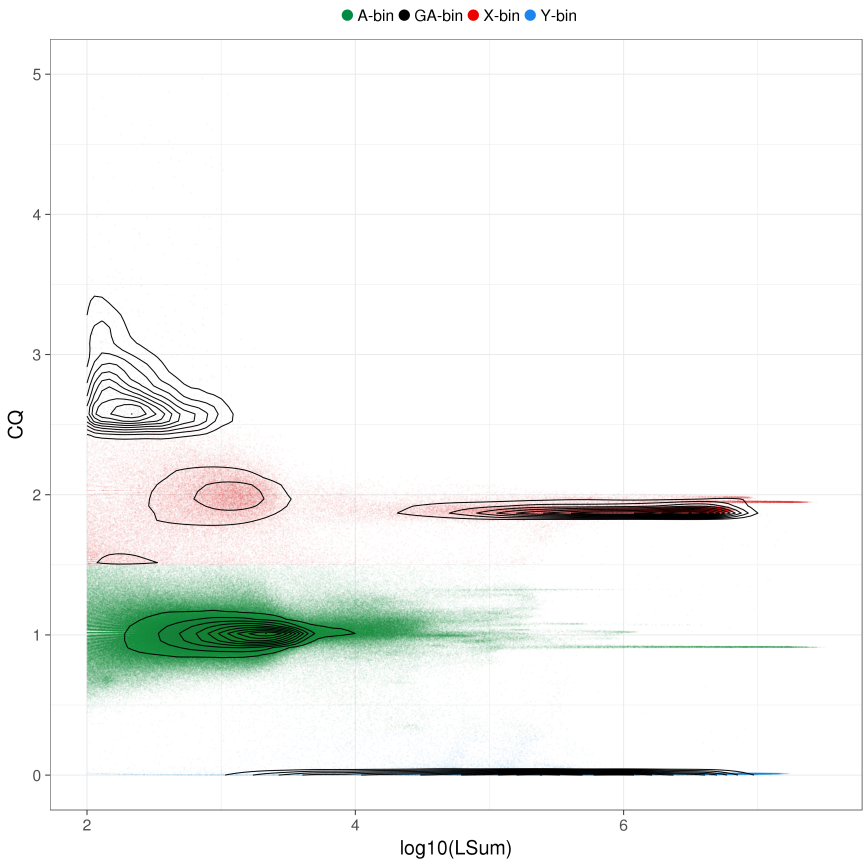
Supplementary Figure 5 - redkmer_plot_reads_4: Histogram of PacBio read log10(sum) distribution.



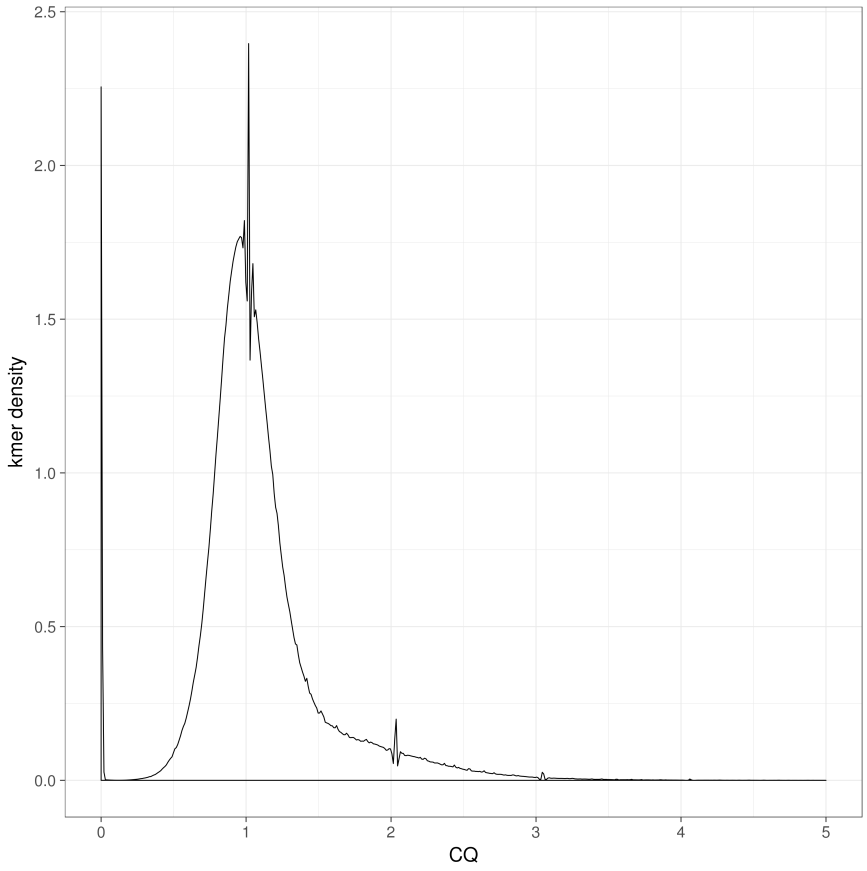
Supplementary Figure 6 - redkmer_plot_reads_5: Histogram of PacBio read length distribution.



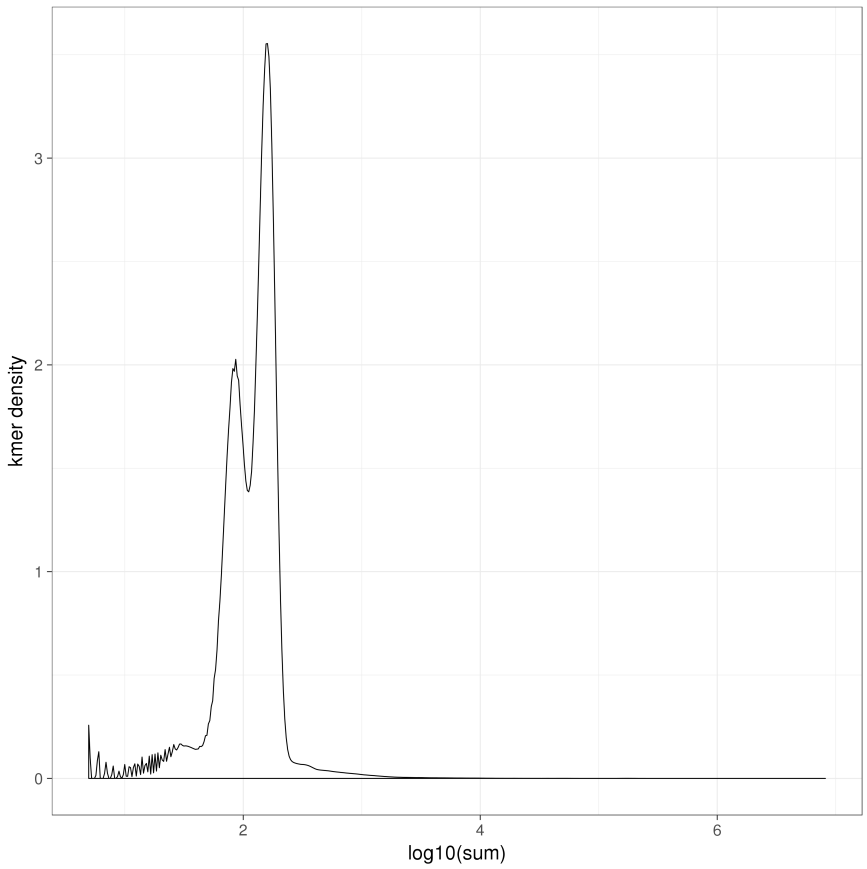
Supplementary Figure 7 - redkmer_plot_reads_6: PacBio read log2(LSum) boxplots for each chromosomal bin with median log2(LSum) for each bin printed below.



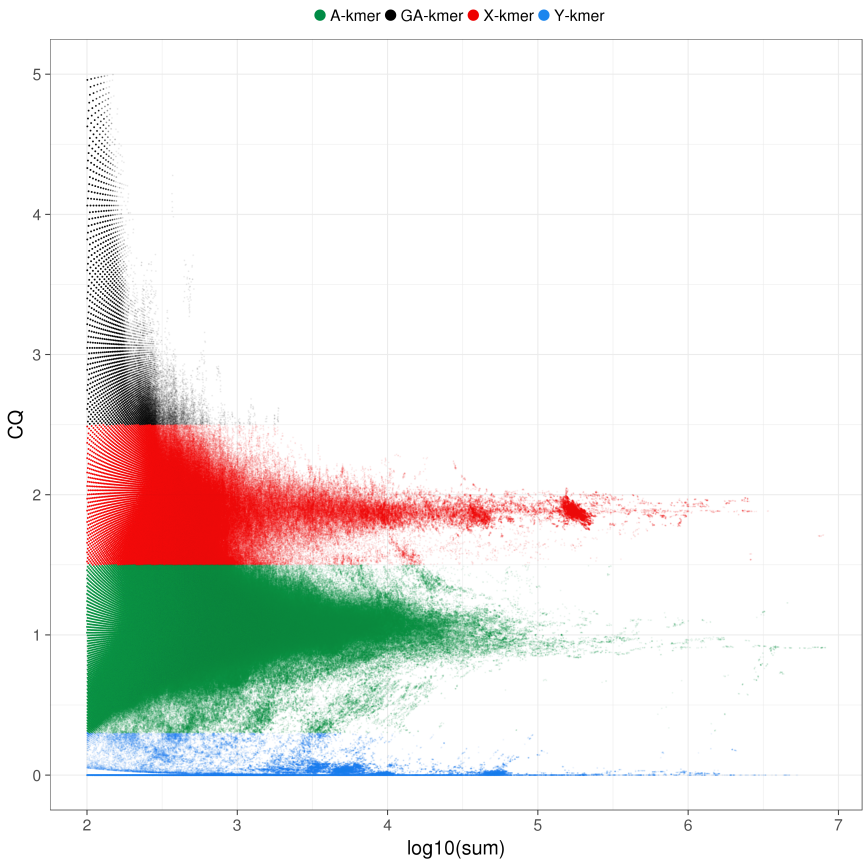
Supplementary Figure 8 - redkmer_plot_reads_7: PacBio read CQ over log10(LSum) with per chromosomal bin data density (in tenth percentile bins).



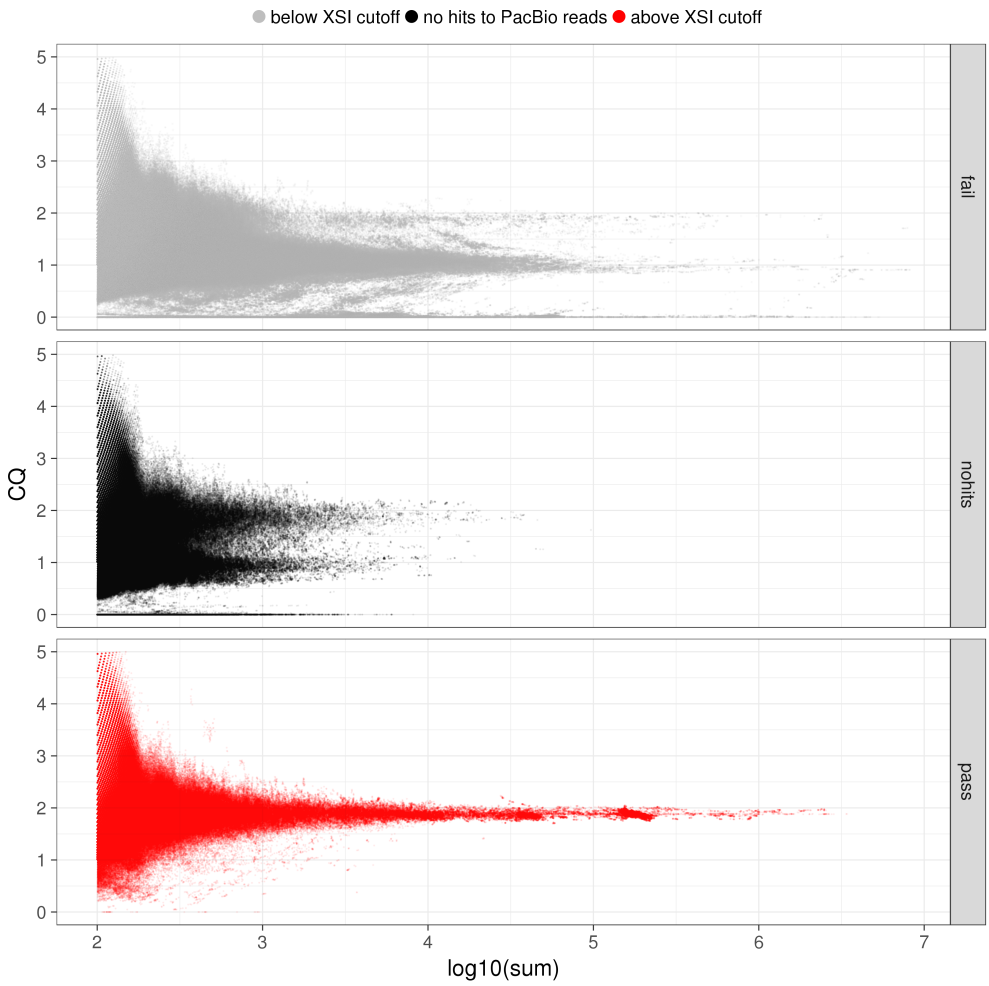
Supplementary Figure 9 - redkmer_plot_kmers_1: Density plot of kmer CQ distribution.



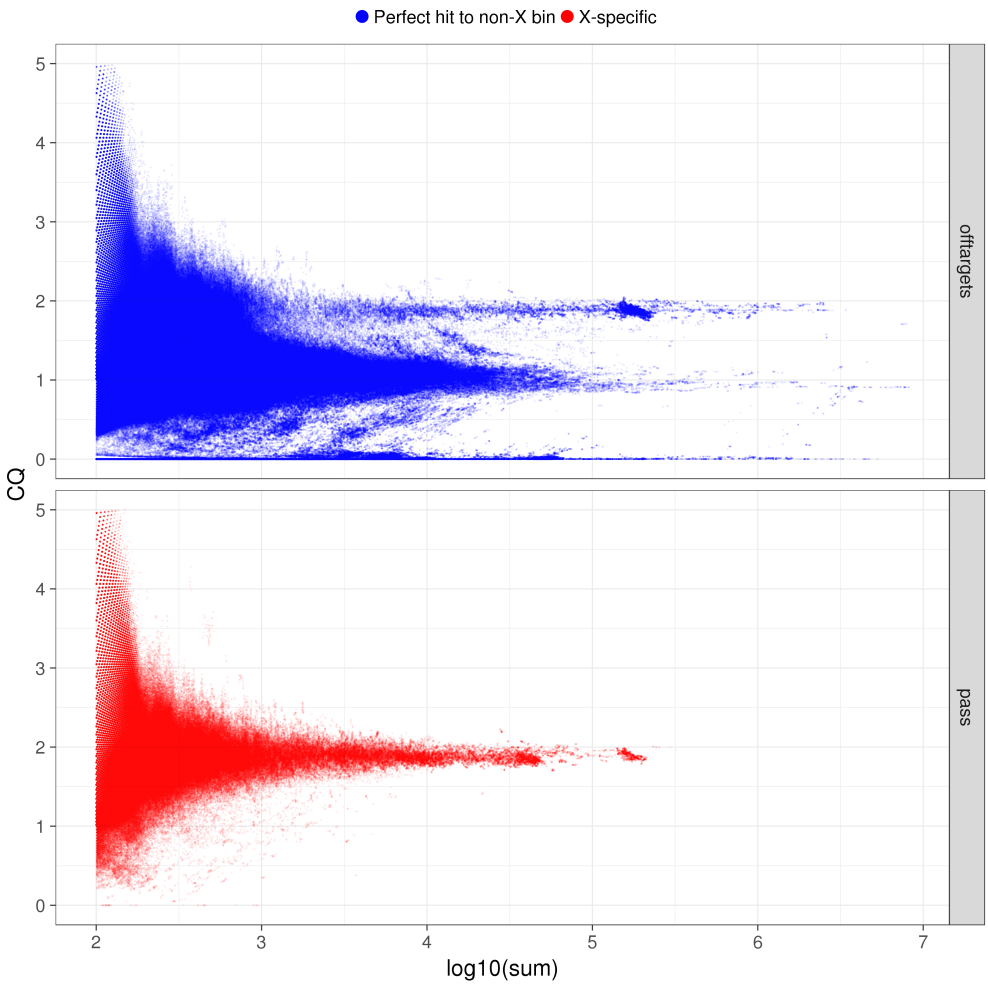
Supplementary Figure 10 - redkmer_plot_kmers_2: Density plot of kmer log10(sum) distribution



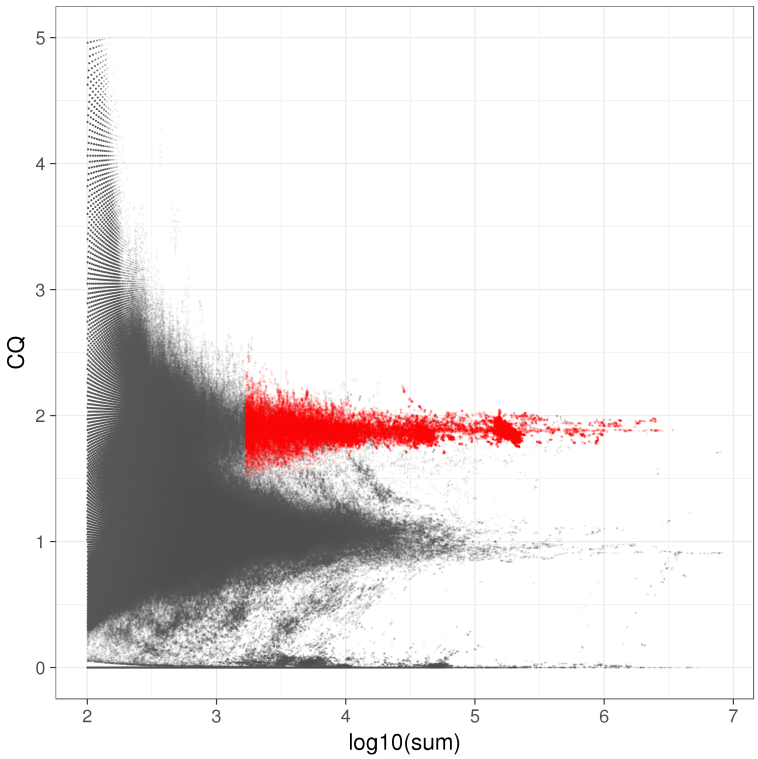
Supplementary Figure 11 - redkmer_plot_kmers_3: kmer CQ (ratio of female/male data) over log10(sum) (total number of kmer occurrences from male and female in illumina data) for all kmers.



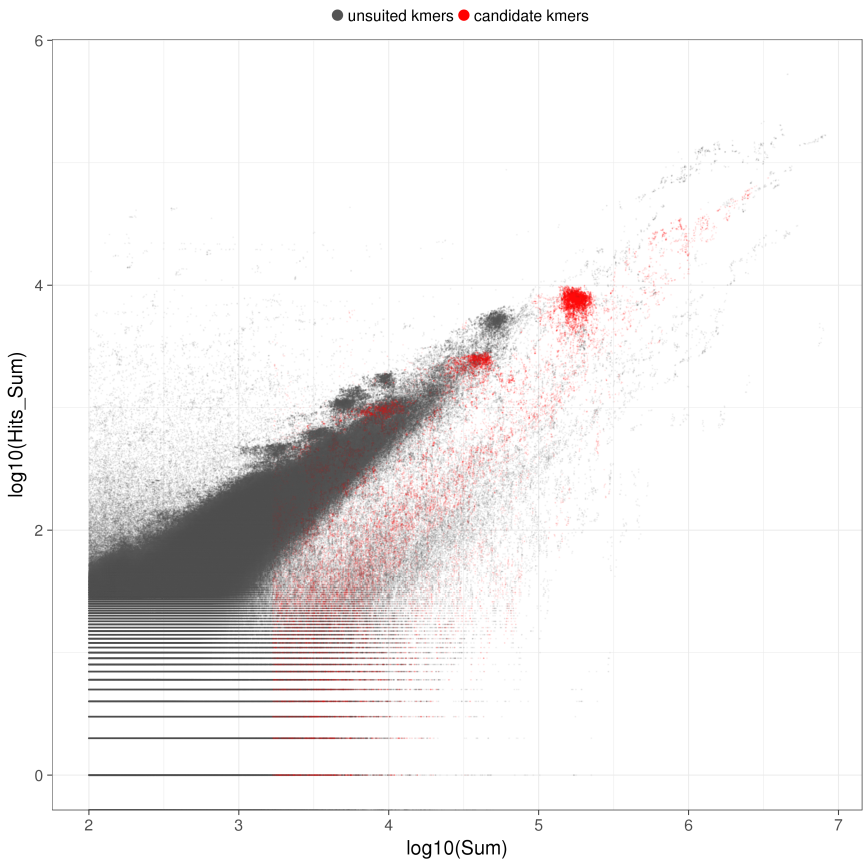
Supplementary Figure 12 - redkmer_plot_kmers_4: kmer CQ over log10(sum) with XSI thresholds.



Supplementary Figure 13 - redkmer_plot_reads_5: kmer CQ over log10(sum) with off-target classifications



Supplementary Figure 14 - redkmer_plot_reads_5: kmer CQ over log10(sum) with final redkmer selection



Supplementary Figure 15 - redkmer_plot_reads_5: kmer log(sum) - coverage in the Illumina data, over log10(Hits_sum) - coverage in the PacBio data, with final selection of kmers shown in red.

Redkmer modules

1. Quality filtering and control (1_redkmer_QC_mito_bowtie2.bash)

As a first step, Redkmer eliminates reads from the long reads library that are shorter than the minimum length as defined in the settings of *redkmer.cfg* pac_length=in bp]. We recommend a minimum length of 500bp. Redkmer then maps the two short read libraries to the mitochondrial genome using bowtie2 (REF) using default settings and excludes all mapping reads from the final short read libraries. The output files are inserted in the relevant data directories and called m_pac.fasta, m.fastq and f.fastq. The quality of the illumina based short read libraries are checked using FastQC quality control tool and a report is generated for each of the two libraries in the *QualityReports* folder.

2. Mapping short to long reads (2_redkmer_ill2pac_bowtie1.bash)

To improve performance, the PacBio fasta file is divided into chunks as a function of the number of nodes made available to Redkmer [NODES= # of nodes]. Illumina reads are then mapped to each chunk using Bowtie (REF) with the following parameters:

bowtie

-a: reporting all alignments

-t: instruct bowtie to report timing of mapping

-5 [TRIMM5]: length in bp to trim at the 5' end of illumina reads

-3 [TRIMM3]: length in bp to trim at the 3' end of illumina reads

-p [cores]: number of cores to be used by each node

-v 0: accept no mismatches

index_m_pac: bowtie index of each length filtered PacBio library chunk

–**suppress** 1,2,4,5,6,7,8,9: report only the ID of the mapped PacBio read in output

m/f.fastq: fastq file (either male or female library)

1> male/female.txt: output file containing ID of mapping hits

2> /pacBio_illmapping/logs/index_array_male/female_log.txt: log file

where the parameters [TRIMM5], [TRIMM3] and [cores] are set in the *redkmer.cfg* file.

The output file from each bowtie alignment, containing in each line the ID of the mapped PacBio read, is then processed by the mapsort count program

(located in Cscripts folder) that sorts and counts each line providing for each unique PacBio ID a number of hits from the male or female illumina library. The output files from the sorting step of each chunk are named male_uniq or female_uniq, contain the suffix of the chunk_ID and are located in the *pacBio_illmapping/mapping_rawdata* folder.

3. Chromosomal binning (3_redkmer_pacbins.bash)

Step3 processes the count data, combines the output from each chunk of the PacBio data from Step2 and re-sorts the entire alignment output. The output files from this sorting step are named male_uniq or female_uniq and are located in the *pacBio_illmapping/mapping_rawdata* folder. The independent male and female outputs are merged and read counts are normalized to library size. Step 3 then generates the *pacBio_MappedReads.txt* file that provides for each PacBio read the following information:

length: PacBio read length

female: number of normalized mapping reads in the female library

male: number of normalized mapping reads in the male library

CQ: ratio of mapping reads (female/male also known as chromosomal quotient (REF))

Sum: sum of mapping reads from both male and female libraries

LSum: length normalized sum of mapping reads (sum of mapped reads / length of read x median length of all reads)

PacBio reads are then filtered based on the total number of length normalized mapping reads (LSum), defined in the *redkmer.cfg* [LSum] and we recommend a minimum LSum of 50. PacBio reads with mapping reads lower than the LSum defined in *redkmer.cfg* are not reported in the final output or used to generate the chromosomal bins.

Four chromosomal "bins", predicting the chromosomal origin of PacBio reads based of the ratio of mapping Illumina reads are then generated: autosomal, X, Y chromosome and "GA". The "GA" bin represents cases of possible genome amplification that may occur in the germline tissues of females, as is known for example in *Drosophila* (REF). The binning based on mapping ratio is performed using three variables, namely *xmin*, *xmax* and *ymax* which are set in the *redkmer.cfg* file. We usually recommend *xmin* of 1.5, *xmax* of 2.5 and *ymax* of 0.3.

X-bin: mapping ratio $\geq xmin$ $\leq xmax$

A-bin: mapping ratio $\geq ymax$ $\leq xmin$

Y-bin: mapping ratio $\leq ymax$

GA-bin: mapping ratio $\geq xmax$

4. kmer generate (4_redkmer_kmers_jellyfish.bash)

Step4 generates kmers of 25bp (ideally suited length for gRNA design) from the female and male illumina libraries using jellyfish (REF). The variable *kmernoise*, defining the minimum number of kmer occurrences to be included in the output of jellyfish is defined in the *redkmer.cfg* file. We usually recommend *kmernoise* = 5, but that will depend on library depth.

5. kmer processing (5_redkmer_kmers_processing.bash)

Step5 processes the kmer output, generates unique IDs for each, normalizes counts to library size and combines the male and female counts. kmers absent in the male library are removed as likely sequencing errors. Step5 then generates the *kmer_counts* file (saved in the *kmers/rawdata* folder) that provides for each kmer the following information:

kmerID: a unique ID defining each kmer

seq: kmer sequence (25bp)

female: normalized kmer occurrences in the female illumina library

male: normalized kmer occurrences in the male illumina library

CQ: ratio of occurrences (female/male also known as chromosomal quotient (REF))

Sum: sum of occurrences from both male and female libraries

All kmers present in the *kmer_counts* file are also exported as a fasta file called *allkmers.fasta* (saved in the *kmers/fasta* folder).

6. Mapping kmers to long reads (6_redkmer_kmers2bins_bowtie1.bash)

To assess chromosome specificity, Step6 of redkmer uses bowtie to align all kmers in the *allkmers.fasta* file against each of the four chromosomal bins generated in Step3. To improve performance, the chromosomal bins are divided into chunks as in Step3. Mapping is performed using the following parameters:

bowtie

-a: reporting all alignments

-t: instruct bowtie to report timing of mapping

-p [cores]: number of cores to be used by each node

–**large-index** instructs bowtie that the index has been generated using the large-index option

-v 0: accept no mismatches

chunk{BINNAME}: bowtie index of each chunk of each bin

–**suppress** 2,3,4,5,6,7,8,9: report only the ID of the kmer

-f allkmers.fasta: fasta file input for mapping

1> {BINNAME}.txt: output file containing ID of mapping kmers

2> chunk_{BINNAME}_log.txt: log file

where the parameter [cores] are set in the *redkmer.cfg* file and BINNAME can be one of the four possible chromosomal bins (X,A,Y or GA)

The output file from each bowtie alignment, containing in each line the ID of the mapped kmer, is then processed by the mapsort count program (located in Cscripts folder) that sorts and counts each line providing for each unique kmer ID, a number of hits to each chromosomal bin chunk. The output files from the sorting step of each chunk are named chunk_kmer_hits_BINNAME and are located in the *kmers/bowtie/mapping* folder.

7. kmers-2-bin processing (7_redkmer_kmers2bins_processing.bash)

Step7 processes the kmer mapping data, combines the output from each chunk of each chromosomal bin from Step6 and re-sorts the entire alignment output for each bin. The output files from this sorting step are named kmer_hits_{BINNAME} and are located in the *kmers/bowtie/mapping* folder. The four outputs for each bin are merged and combined with the *kmer_counts* file from Step5. The "X-specificity index" (XSI) is then calculated by diving the number of hits to the X-chromosome bin by all hits to all bins. If the XSI is higher or equal to the XSI defined in the *redkmer.cfg*, kmers are labelled as "pass", otherwise "fail". We recommend an XSI of 0.9, i.e. at least 90% of hits have to occur on the X chromosome. If the are no hits to the chromosomal bins, then kmers are labelled with "no hits". The output of Step7, the *kmers_hits_results* file locate in the *kmers/rawdata* folder, provides for each kmer the following information:

kmerID: a unique ID defining each kmer

seq: kmer sequence (25bp)

female: normalized kmer occurrences in the female illumina library

male: normalized kmer occurrences in the male illumina library

CQ: ratio of occurrences (female/male also known as chromosomal quotient (REF))

Sum: sum of occurrences from both male and female libraries

hits_X: number of hits to the X-chromosome bin

hits_A: number of hits to the autosome bin

hits_Y: number of hits to the Y-chromosome bin

hits_GA: number of hits to the GA-chromosome bin

hits_sum: total number of hits to all chromosomal bins

percenthitsX: number of hits_X / hits_sum

hits_threshold: threshold for *percenthitsX* where:

if $\geq$ XSI = "pass"

if $<$ XSI = "fail"

if no hits to chromosomal bins = "no_hits"

All kmers with a *hits_ theshold* == "pass" are extracted in the *Xkmers.fasta* file in the *kmers/fasta* folder and used in the subsequent steps.

8. Off-targets mapping (8_redkmer_kmers2bins_off_bowtie1.bash)

To assess the potential for off-targeting - illegitimate cutting of non-canonical but related sequences by the endonucleases, Step8 uses bowtie to map candidate X-kmers from the *Xkmers.fasta* file against the A, Y and GA chromosomal bins (i.e. all but the X-chromosome bin) by ***allowing mismatches in the alignment in up to 2 of the 25bp*** (80% identity). Again, to improve performance, chromosomal bins are divided into chunks as in Step3 and Step6. Mapping is performed using the following parameters:

bowtie

-a: reporting all alignments

-t: instruct bowtie to report timing of mapping

-p [cores]: number of cores to be used by each node

–**large-index** instructs bowtie that the index has been generated using the large-index option

-v 2: accept up to 2 mismatches in alignment

chunk{BINNAME}: bowtie index of each chunk of each bin

–**suppress** 2,3,4,5,6,7,8,9: report only the ID of the kmer

-f allkmers.fasta: fasta file input for mapping

1> {BINNAME}.txt: output file containing ID of mapping kmers

2> chunk_{BINNAME}_log.txt: log file

where the parameter [cores] are set in the *redkmer.cfg* file and BINNAME can be one of the three possible chromosomal bins (A,Y or GA)

The output file from each bowtie alignment, containing in each line the ID of the mapped kmer, is then processed by the mapsort count program (located in Cscripts folder) that sorts and counts each line providing for each unique kmer ID, a number of hits to each chromosomal bin chunk. The output files from the sorting step of each chunk are named chunk_kmer_hits_BINNAME and are located in the *kmers/bowtie/offtargets* folder.

9. Off-targets processing (9_redkmer_kmers2bins_off_processing.bash)

Processing of offtargets is performed as in Step7, except by the lack of X-chromosome bin data (as off-targeting here can be accepted) and that the output, in number of hits to the chromosomal bins (here A,Y and GA), are combined and reported in a single column (i.e. not separately as in Step7). The output from this step is the *kmer_ results.txt* file in the *kmers* folder and reports the following data:

kmerID: a unique ID defining each kmer

seq: kmer sequence (25bp)

female: normalized kmer occurrences in the female illumina library

male: normalized kmer occurrences in the male illumina library

CQ: ratio of occurrences (female/male also known as chromosomal quotient (REF))

Sum: sum of occurrences from both male and female libraries

hits_X: number of hits to the X-chromosome bin

hits_A: number of hits to the autosome bin

hits_Y: number of hits to the Y-chromosome bin

hits_GA: number of hits to the GA-chromosome bin

hits_sum: total number of hits to all chromosomal bins

percenthitsX: number of hits_X / hits_sum

hits_threshold: threshold for *percenthitsX* where:

if $\geq$ XSI = "pass"

if $<$ XSI = "fail"

if no hits to chromosomal bins = "no_hits"

sum_offtargets: number of hits to canonical and illegitimate hits to non-X chromosomal bins

offtargets: number of canonical (100% identity hits) to non-X chromosomal bins

degen_targets: number of non-canonical, degenerate (up to 80% identity hits_ to non-X chromosomal bins ***(true offtargets)***

10. Final processing and X-kmer selection (10_redkmer_kmers_output.bash)

Step10 performs several calculations and annotations of the above mapping data to select the most specific and the most abundant Xkmers. First, thresholds of kmer CQ (occurrence ratio in female/male data) are used to annotate chromosomal origin using three variables, namely *kmer_xmin*, *kmer_xmax* and *kmer_ymax,* which are set in the *redkmer.cfg* file and annotations are reported in the *candidate* column of the output. These are different from the previously used *xmin*, *xmax* and *ymax* variable, which are used from binning of PacBio reads. We usually recommend *kmer_xmin* of 1.5, *kmer_xmax* of 2.5 and *kmer_ymax* of 0.3.

X: mapping ratio $\geq$*kmer_xmin* $<$*kmer_xmax*

A: mapping ratio $\geq$*kmer_ymax* $\leq$*kmer_xmin*

Y: mapping ratio $\leq$*kmer_ymax*

GA: mapping ratio $\geq$*kmer_xmax*

Redkmer then defines the "*selection*" column that annotates kmers as "good kmers" if:

XSI_threshold: == pass

kmer_CQ: == X

kmer_abundance: to be within 99.5 percentile of Xkmer occurrence *(Sum)*

Otherwise the selection variable for each kmer is annotated as "bad kmer". Step10 then outputs all kmer data into the *kmer_results.txt* file, and the selected Xkmer data into candidateXkmers.txt and the candidateXkmers.fasta files, all of which are saved in the *kmers* folder. The *kmer_results.txt* file is then cut into 7 smaller files containing information needed for plotting in the *kmers/dataforplotting* folder.