Supplementary Document for
QuicK-mer: A rapid paralog sensitive CNV detection pipeline

Table of Contents

Defining a Catalog of Unique K-mers

Defining a catalog of unique k-mers requires 7 individual steps. In practice, we utilize a size of k=30 for consistency with previous studies (Sudmant *et al.*, 2010; Alkan *et al.*, 2009). An example of the command lines for each of the following steps can be found in the QuicK-mer User Manual v1.0 Section 9.

1. *List all unique 30-mers* All 30-mers in the reference genome are enumerated with Jellyfish by setting k-mer size equal to 30 and using the reference genome FASTA sequence as the input. A k-mer and its reverse complement are considered equal (Jellyfish option –C). The 30-mers with a count of 1 are exported into text format.

2. *Determine unique 30-mer locations* Unique 30-mers are mapped to the genome reference using mrsFAST (Hach *et al.*, 2010) with an edit distance setting of 0. This step serves to map the location of each unique 30-mer and is used in the following steps for region overlapping and exclusion.

3. *Enumerate highly repetitive 15-mers* The same procedure for Step 1 is repeated for the reference assembly except now k is set equal to 15 and all 15-mers with counts ≥ 1,000 are exported.

4. *Determine repetitive 15-mer locations and filter 30-mers* Step 2 is repeated with the 15-mers determined in Step 3. Here, each k-mer will have multiple genome locations.

Finally, locations of the 15 and 30-mers are merged together, and all 30-mers (from Step 2) that overlap with the high frequency 15-mer track are removed from subsequent analyses.

5. *Remove highly similar 30-mers* The 30-mers that pass Step 4 are mapped onto the reference genome using mrsFAST with an edit distance of 2. All 30-mers with ≥ 100 mapped positions are removed. Note that mrsFAST only considers substitutions.

6. *Remove highly similar 30-mers, considering indels* The k-mers that pass Step 5 are mapped again using mrFAST with an edit distance of 2. All 30-mers with ≥ 100 mapped positions are removed. The mrsFAST search is performed prior to mrFAST due to the speed advantage of mrsFAST only considering mismatches. Steps 5 and 6 serve to reduce the chances of matching k-mers with sequencing errors into unintended locations.

7. *Combine final k-mer catalog* The final list of highly unique 30-mers is sorted based on chromosome location and output in BED format. This output file will then be used by QuicK-mer and for the generation of required axillary files.

QuicK-mer Core Pipeline

Depth Counting and GC Correction

The QuicK-mer core program is written in C++ and Object Pascal and wrapped with Python for control flow. The control flow consists of calling Jellyfish-2 (Marçais and Kingsford, 2011) for building the 30-mer hash library followed by the k-mer query step. At the beginning of the query step, two axillary binary files are preloaded and memory space for count values is allocated. QuicK-mer then interrogates the Jellyfish hash library with the sorted 30-mer list, storing each raw count value in memory. The core program verifies each 30-mer's status as a normalization control and, if indicated, the 400 bp GC-content value is fetched from the associated binary file and incorporated into the GC bias curve. Once the process is finished, the core program builds the GC curve based on the average counts obtained from each GC percentage bin and uses the lowess smoothing algorithm to generate a correction curve. The targeted average depth is calculated using a weighted average based on GC content of 25~75%. A 0.3x minimum and 3x maximum correction factor is also enforced to reduce over-correcting extreme GC regions due to a lack of representative k-mers. The GC bias curve is output in a text format and, along with correction curve, is represented in a PNG image (S-Fig 2). Lastly, the correction factor is applied to each k-mer count value based on its GC content and the resulting GC-corrected k-mer counts are output in a binary format.

Due to different GC biases within sequencing libraries and across flow cell lanes (S-Fig 3), the user is encouraged to apply QuicK-mer GC normalization separately for each sequencing lane. Resulting GC-corrected k-mer counts can then be merged together for each sample using the CorDepthCombine command.

Normalization Step

Another program in the QuicK-mer package (kmer2window) converts counts to copy-number estimates. The normalization program loads the same binary control region file then, using the corrected depth for the control 30-mers, calculates a scaling factor based on an assumed copy number of two for these regions. Normalization is performed in windows of equal number of k-mers (default = 500 k-mers per window, but is adjustable by the user). The median k-mer count for each window is used for the normalization, and only windows where all k-mers are in the defined control intervals are used in subsequent steps. The resulting normalization is then applied to all windows. Please see the QuicK-mer User Manual for detailed examples and commands.

Supplementary Results

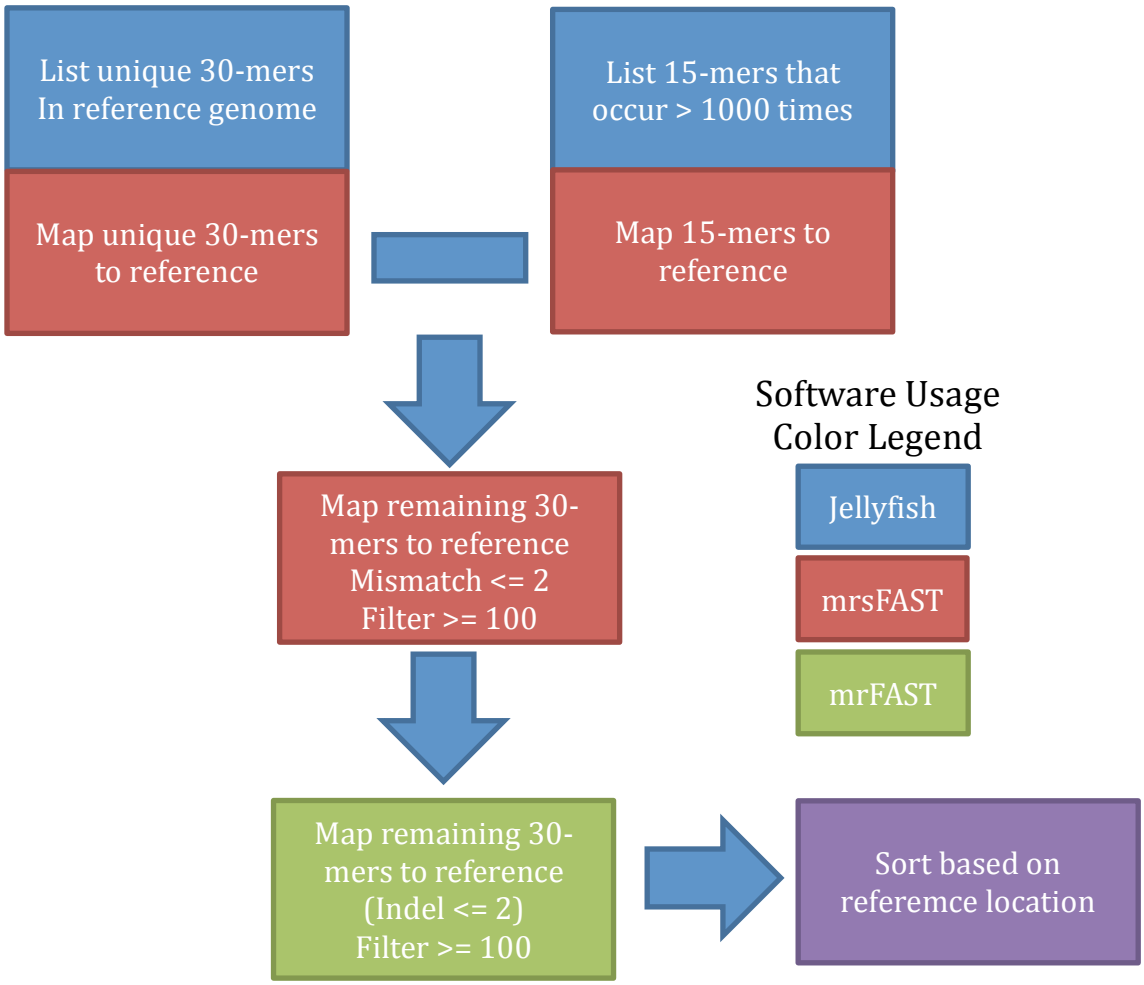
For comparison, we reanalyzed dataset from the 1000 Genome Project and other sources using QuicK-mer and compared the estimated copy number profiles with supplementary data from the Sudmant *et al.*, 2010 paper. The major dataset was downloaded from 1000 Genome Project Pilot, Phase 1 and Phase 3 studies (Abecasis *et al.*, 2010, 2012; Auton *et al.*, 2015). Sequencing files were individually run through the QuicK-mer pipeline and GC corrections were performed for each sequencing lane. Corrected data is combined and normalized into copy number estimates. Table 1 contains the details of samples used in this study.

Supplementary Figures 5 – 14 correspond to the genome regions depicted in S52, S60 – S71 of Sudmant *et al.*, 2010. Regions shown in these figures match with the CNV identified in the original paper and demonstrate the accuracy of QuicK-mer.

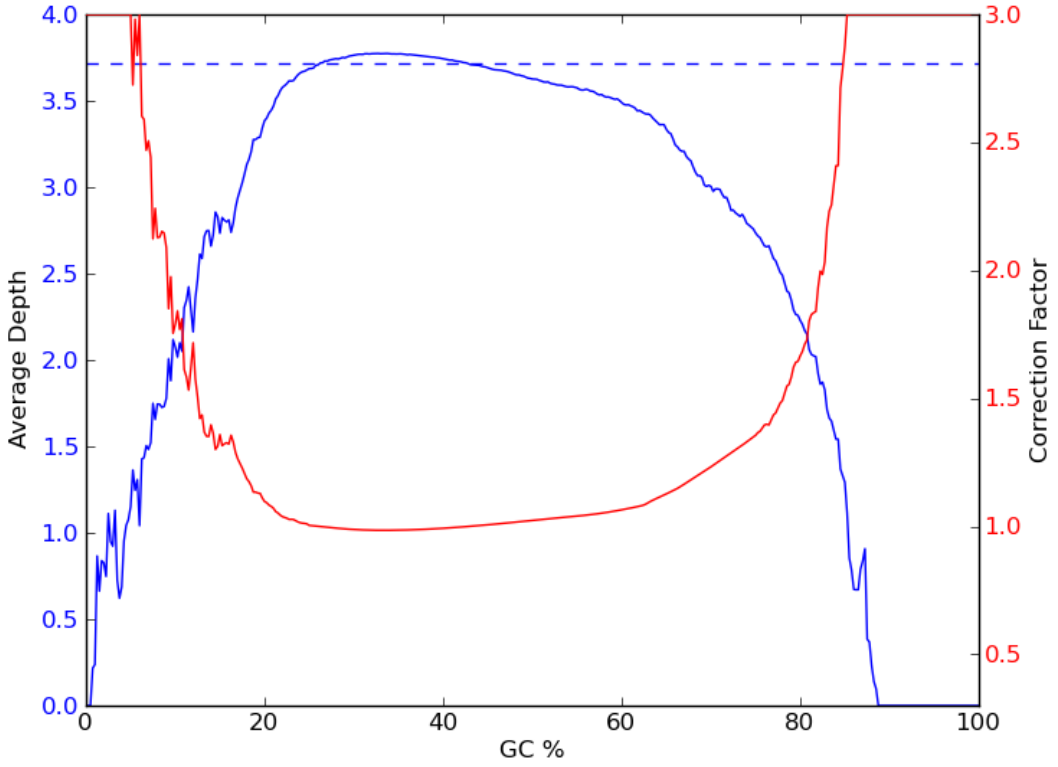
Sample Name	Data Source	Mean 30-mer Depth in Control
NA12156	1000 Genome Phase 3	4.40
NA12878	1000 Genome Phase 1, Pilot 1/2	7.23
NA18507	Bentley et al., Nature 2008	23.05
NA18508	Bentley et al., Nature 2008	7.30
NA18517	1000 Genome Phase 3	3.82
NA18555	1000 Genome Phase 3	4.09
NA18956	1000 Genome Phase 3	3.63
NA19129	1000 Genome Phase 1, Pilot 1/2	0.87
NA19240	SRX574476, SRX582073	13.26

Supplementary Table 1. Samples used in this study
The mean depth is calculated based on the median depth obtained from windows of 500 30-mers fully overlapping with defined control regions.

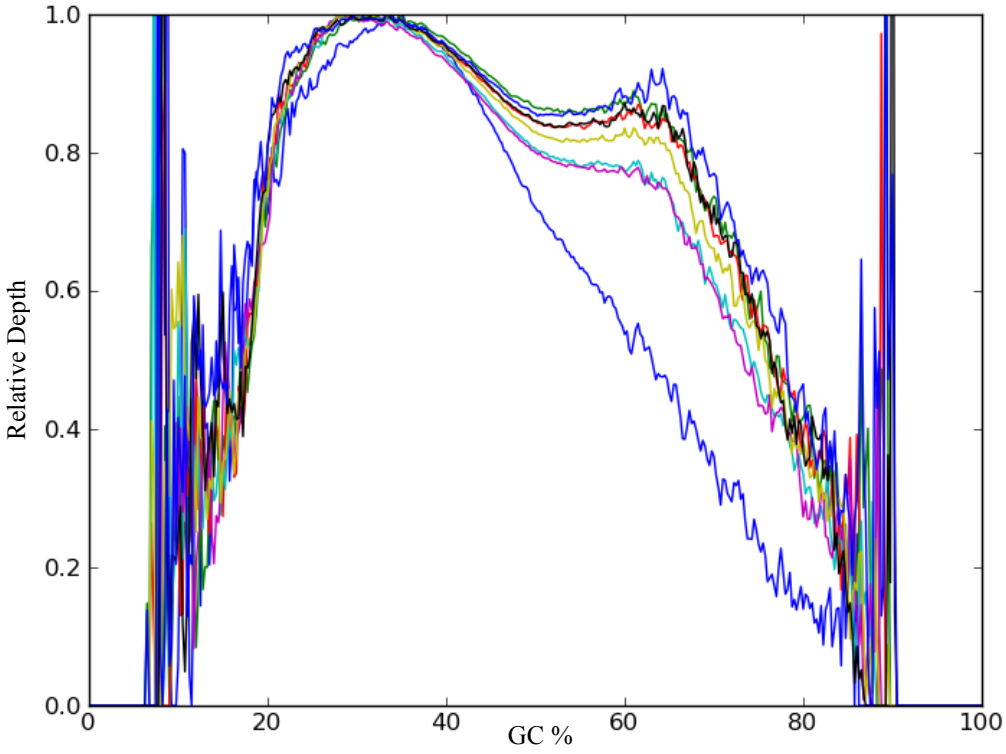
Supplementary Figures



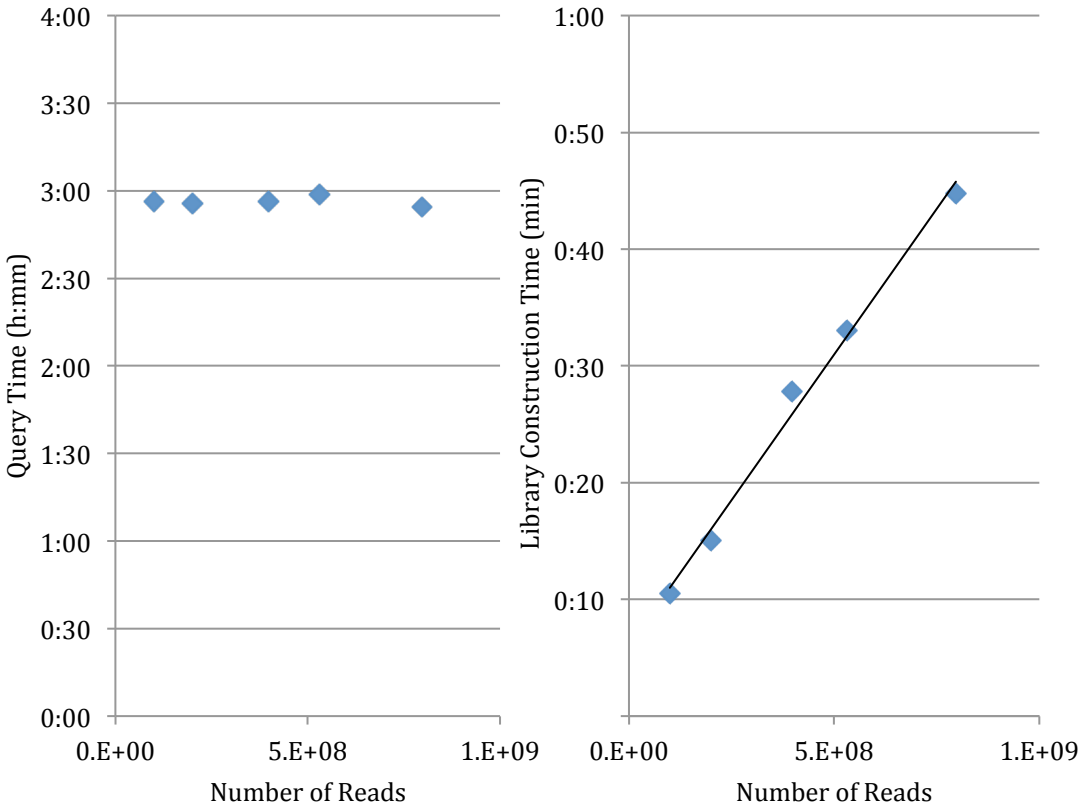
Supplementary Figure 1. Unique 30-mer Workflow
The above workflow illustrates the procedure used to obtain unique 30-mers from a reference genome assembly. Software used in each step are color-coded and detailed in the legend on the right.



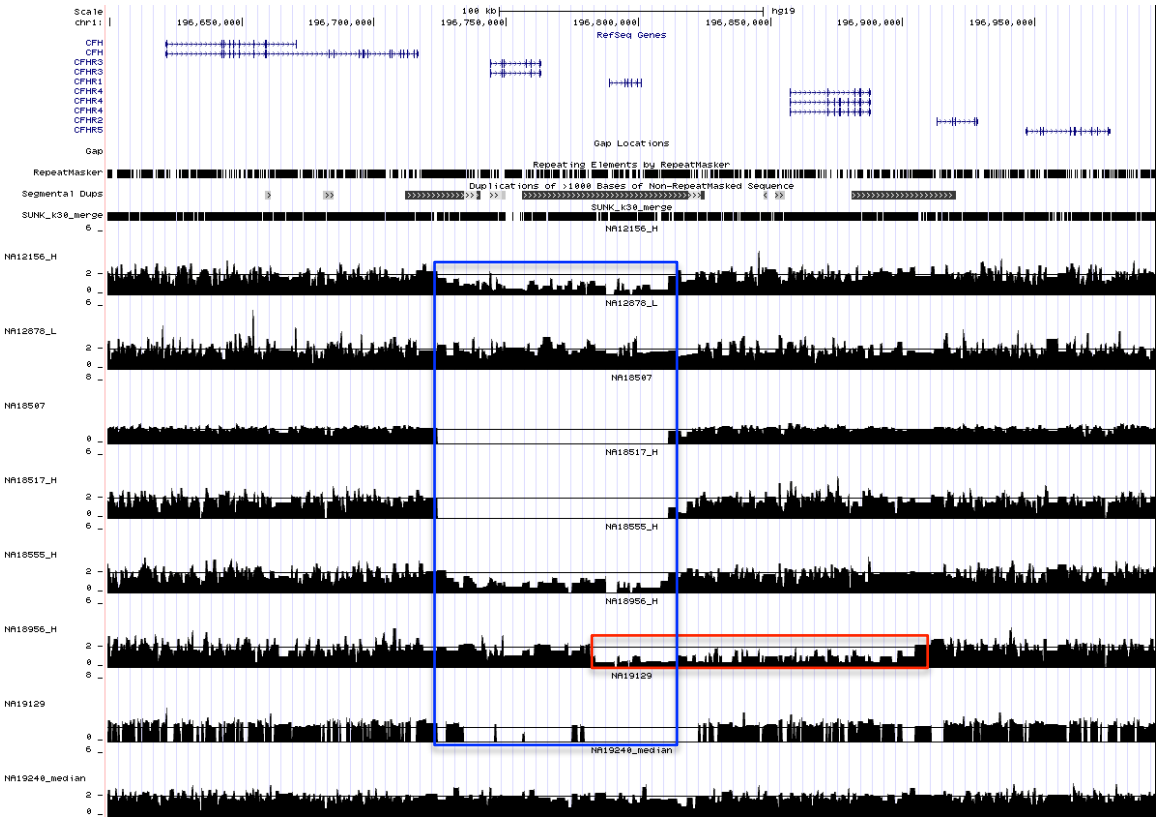
Supplementary Figure 2. GC Correction and Bias
The majority of sequencing coverage bias is related to local GC content. The blue curve indicates the average depth for the 30-mers with the same GC content in 400bp surrounding the center of each k-mer location, rounded in steps of 0.25%. The red curve is the lowess smoothed correction factor, targeted for the average depth indicated by the dashed line. A 3x max correction value is enforced. The GC curve represents QuicK-mer run from WGS experiment SRX734522.



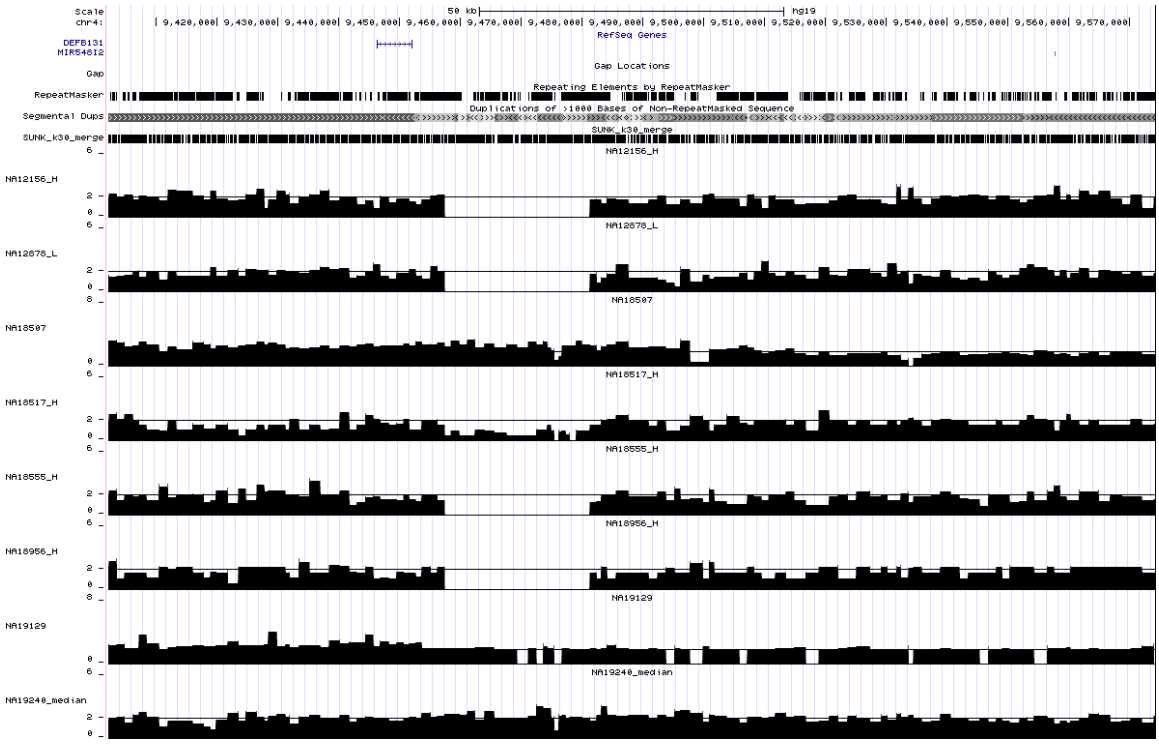
Supplementary Figure 3. GC Bias across sequencing lanes
GC bias can vary between sequencing libraries due to PCR amplification bias. The bias can also be found between sequencing lanes for some PCR-free libraries. Plotted data corresponds to ERX002358 from CAST/EiJ mouse WGS data.



Supplementary Figure 4. Wall time statistics
To assess the efficiency of QuicK-mer, we randomly sampled subsets of reads from HG02799, which was sequenced to a depth of 17x. The selected fractions were individually analyzed using 35GB memory and 4 cores during library construction and 2 cores during querying on an empty compute node with 4 Xeon E7 4850 2GHz processors and 1TB of total memory. Wall clock-time statistics indicate a constant time cost for the querying step once the average sequencing depth exceeds 1x. The nature of counting predefined k-mers also means the memory usage is unlikely to be affected by the sequencing depth. The library building time is linearly correlated with the input read counts.

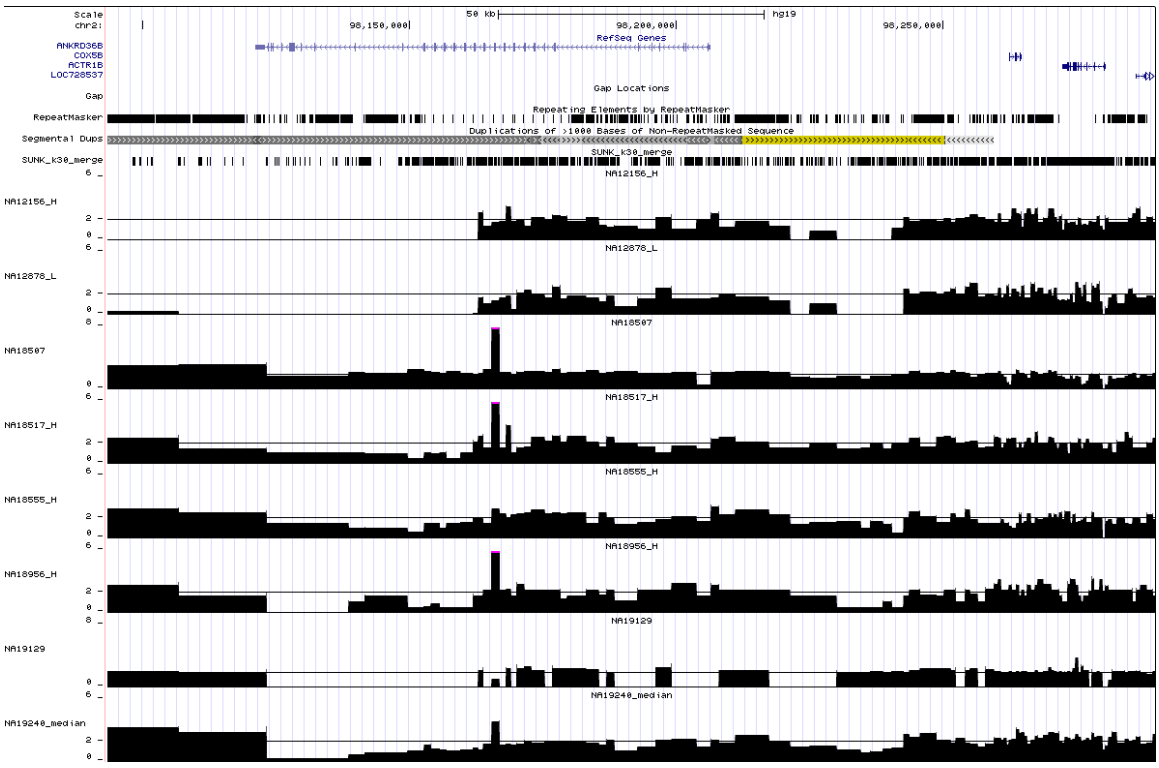


Supplementary Figure 5
Genome browser screenshot that details two overlapping deletions found in several individuals and match with S60 in Sudmant *et al.,* 2010.

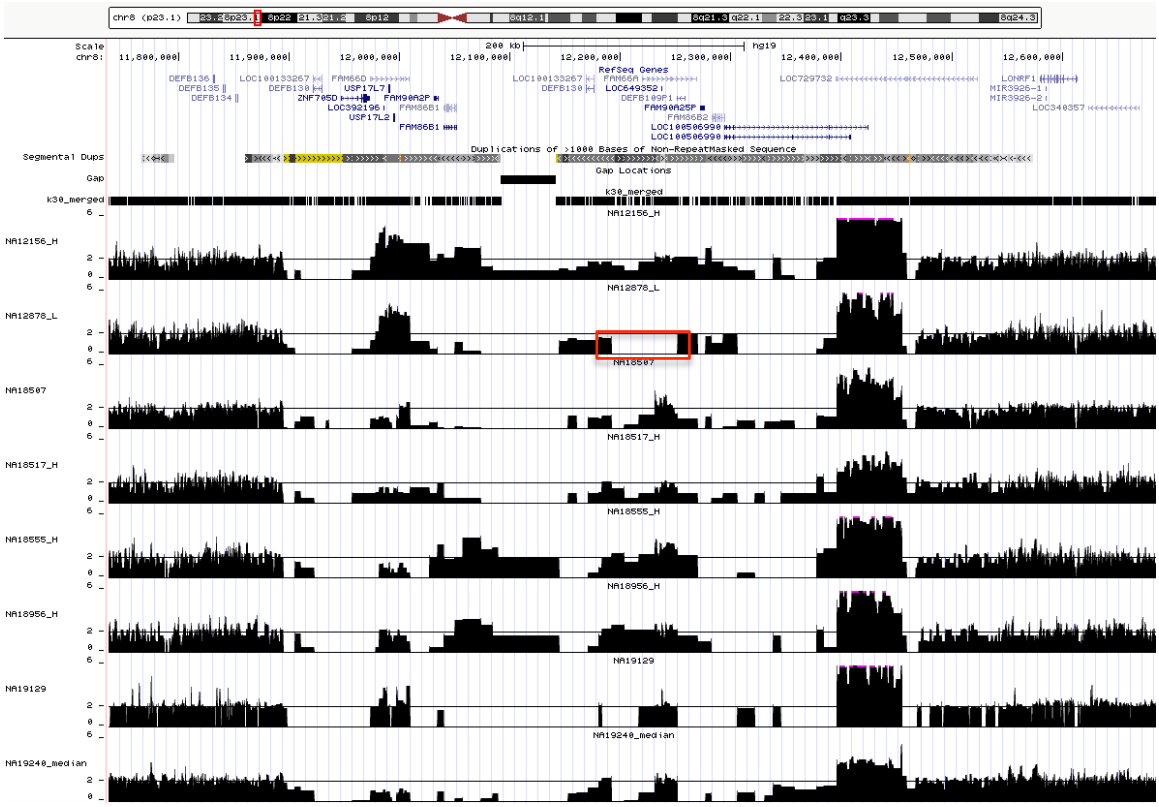


Supplementary Figure 6
Genome browser screenshot that illustrates deletions detected in 5 individuals and a hemizygous deletion in NA18517. This figure corresponds to S61 in Sudmant *et al.*, 2010.

Supplementary Figure 7
Hemizygous deletion of the *PSG* gene cluster at 19q13.31 in NA18956 within the highlighted region. This figure corresponds to S62 in Sudmant *et al.*, 2010.

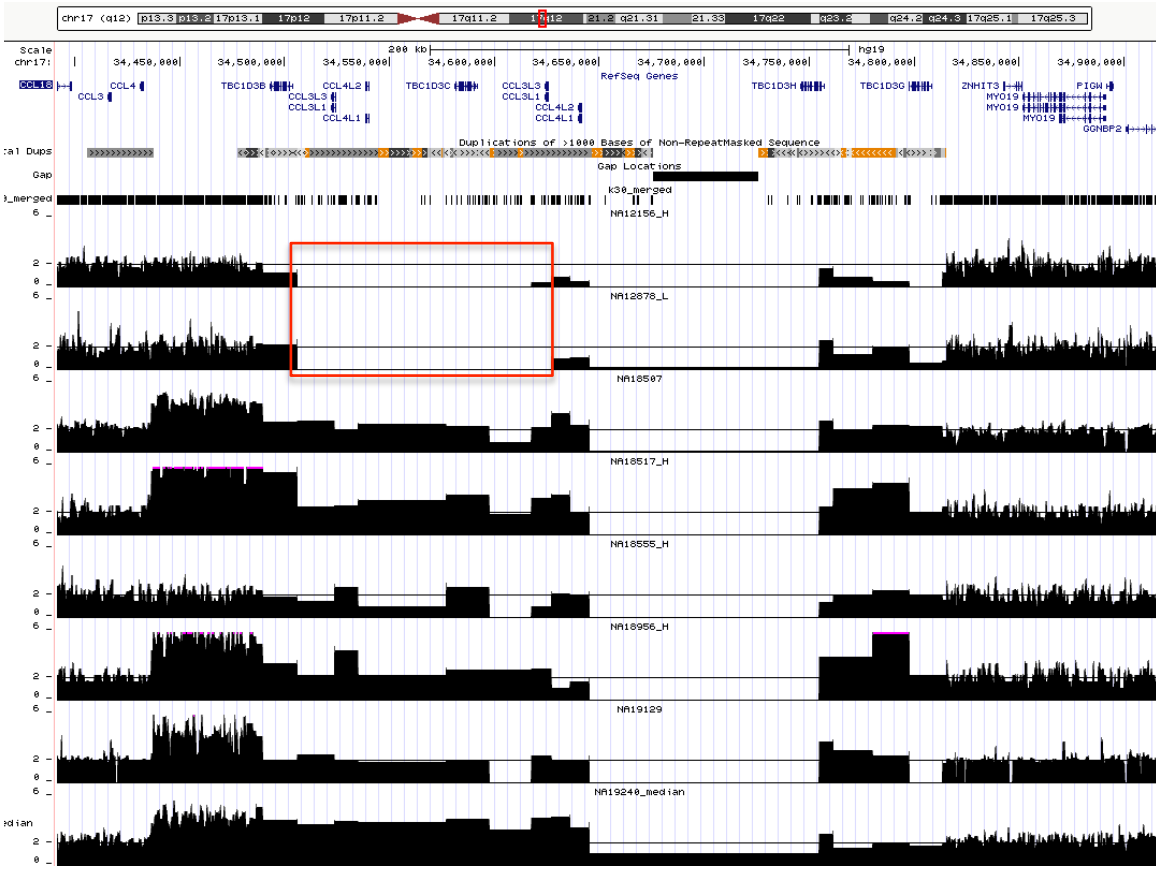


Supplementary Figure 8
Deletion of the 3' regions of *ANKRD36B* in NA12156 and NA12878, which matches the prediction in S63 in Sudmant *et al.,* 2010.



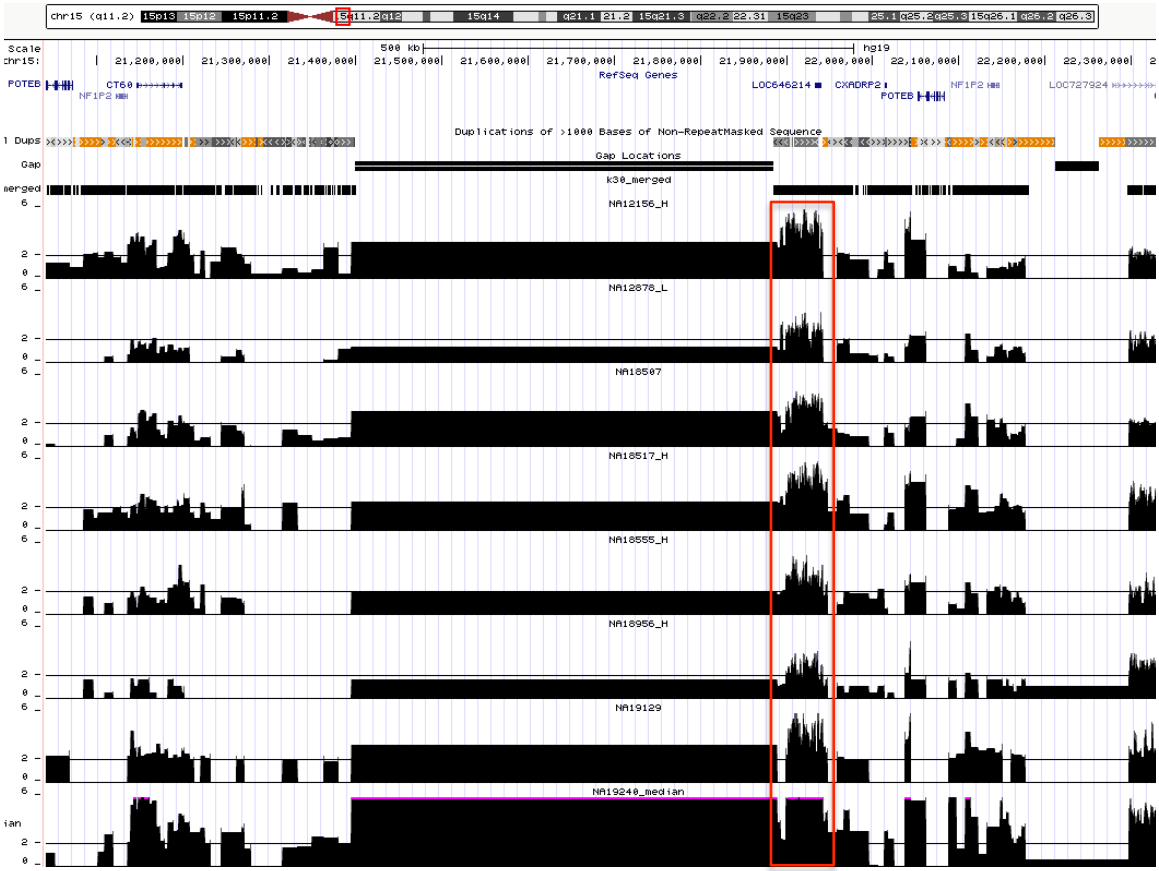
Supplementary Figure 9

This figure corresponds to S66 in Sudmant *et al.,* 2010 and illustrates a deletion of a defensin gene (observed in NA12878).

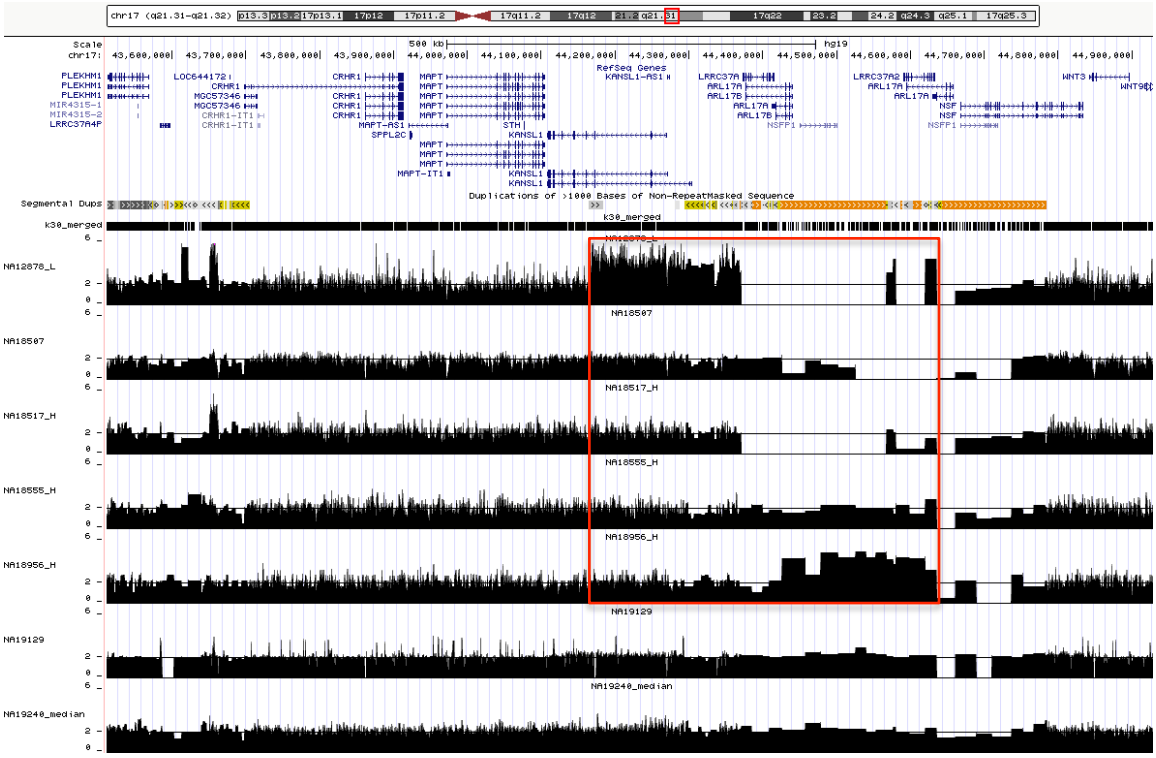


Supplementary Figure 10

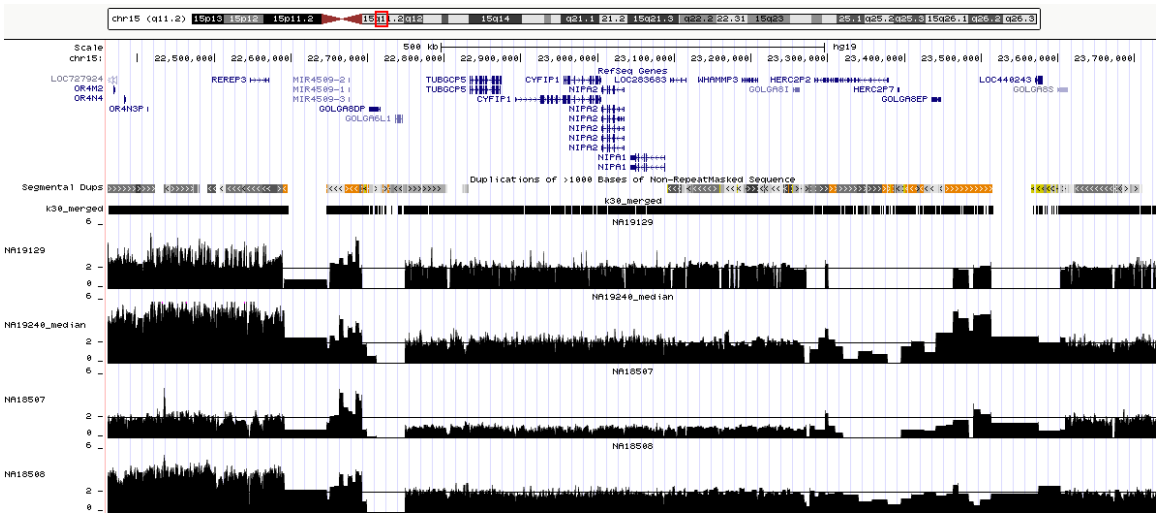
This figure corresponds to S67 in Sudmant *et al.*, 2010. Here, a deletion of chemokine ligand genes (see RefSeq gene IDs in top track) is observed in NA12156 and NA12878.



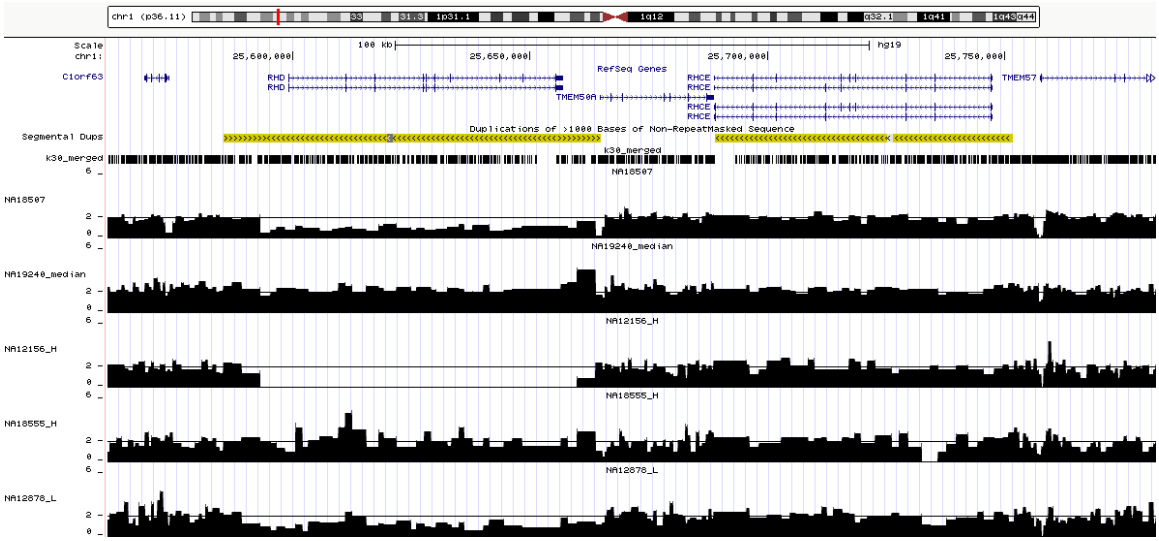
Supplementary Figure 11
This figure corresponds to S68 in Sudmant *et al.*, 2010. The center window overlaps with an assembly gap and the actual copy number is invalid. An adjacent duplication is observed in all samples shown. NA19240 also exhibits duplication in the olfactory receptor gene clusters (see RefSeq gene IDs in top track).



Supplementary Figure 12
This figure corresponds to S69 in Sudmant *et al*., 2010 and displays segmental duplications in the17q21.31 *MAPT* region.



Supplementary Figure 13
This figure corresponds to S70 in Sudmant *et al*., 2010 and illustrates several gene deletions at 15q11.2 in NA18507.



Supplementary Figure 14
This figure corresponds to S71 in Sudmant *et al*., 2010 and shows copy number variation at the *RHD* and *RHCE* genes. NA18507 and NA12878 each have one copy of RHD deleted.

Supplementary References

Abecasis,G.R. *et al.* (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature*, **491**, 56–65.
Abecasis,G.R.R. *et al.* (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–73.
Alkan,C. *et al.* (2009) Personalized copy number and segmental duplication maps using next-generation sequencing. *Nat. Genet.*, **41**, 1061–7.
Auton,A. *et al.* (2015) A global reference for human genetic variation. *Nature*, **526**, 68–74.
Hach,F. *et al.* (2010) mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nature methods*, **7**, 576–577.
Marçais,G. and Kingsford,C. (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, **27**, 764–70.
Sudmant,P. *et al.* (2010) Diversity of Human Copy Number Variation and Multicopy Genes. *Science*, **330**, 641–646.

QuicK-mer v1.0 User Manual

Commands in 4 - 6 assume you start in the root directory of QuicK-mer.

1. Prerequisites
   Before using the QuicK-mer CNV pipeline, here is a list of programs required:
   1) Jellyfish 2
   2) Python 2.7
   3) matplotlib 1.1.0 or later
   4) samtools (only necessary if input file is in BAM format)

2. What is QuicK-mer?
   QuicK-mer is an efficient, paralog-sensitive CNV estimation pipeline based around Jellyfish-2. It counts the occurrences of each predefined k-mer inside Illumina sequencing data and normalizes to correct copy number based on pre-defined control regions. QuicK-mer supports both FASTQ and BAM format as input.

3. Download QuicK-mer
   QuicK-mer is distributed as a source package on github. Grab QuicK-mer using the following command:

   ```
   git clone https://github.com/KiddLab/QuicK-mer.git
   ```

4. Compile
   There are 3 required executables written in a compiled language to increase pipeline efficiency. Pre-compiled binaries are included in the distribution. If an OS/CPU not supported by the existing distributed binary is used, the programs should be compiled by the user.
   1) **KmerCor**
      This is the core program for GC bias estimation and depth normalization in QuicK-mer. To compile use the below command:

      ```
      cd kmer/
      fpc -O KmerCor.lpr
      ```

   2) **kmer2window**
      This program is used to convert depth data into copy number in a bedGraph format based on predefined window sizes and control regions. Each window contains a fixed number of k-mers. Note that the last window at the end of each chromosome may contain fewer.

      ```
      cd kmer/
      g++ -O -o kmer2window kmer2window.cpp
      ```

   3) **CorDepthCombine**
      The CorDepthCombine program is used to merge each GC-corrected sequencing library (or sequencing lane) from the same sample together. Each sequencing

library (or lane) usually contains distinctive GC bias patterns and should be run through QuicK-mer separately.

```
cd kmer/
fpc -O CorDepthCombine.lpr
```

5. Installation

QuicK-mer does not need to be installed, all you need to do is add the application folders to your path directory.

```
QuicK-mer/
QuicK-mer/kmer/
```

To do so in unix-like systems, open your .bashrc file in the home directory using a text editor or with vi. Add the following line:

```
PATH=$PATH: path_before_Quick-mer/QuicK-mer/:path_before_Quick-mer /QuicK-mer/kmer/
```

Then execute using:

```
source .bashrc.
```

6. Premade 30-mer lists available for download

The following genomes have unique 30-mer catalogs ready for download (http://kiddlabshare.umms.med.umich.edu/public-data/QuicK-mer/Ref/):

1) mm10
2) hg19
3) panTro4
4) canFam3.1

7. Description of supporting files

Once extracted, each folder contains six files to support the QuicK-mer pipeline. Using hg19 as an example, below is a list of the six files.

```
hg19_kmer.bed
k30_hg19_GC.bin
k30_hg19_CN2.bin
hg19_50_window.bed
hg19_500_window.bed
hg19_uniq.bc
```

hg19_kmer.bed is the predefined 30-mer list in bed format. It contains the location of each 30-mer and its sequence in the last column. k30_hg19_GC.bin is the GC content of the surrounding 400bp with the 30-mer in the center. k30_hg19_CN2.bin records a true/false flag with each byte per 30-mer indicating if the 30-mer is **excluded** from control region. Hence, 0x00 30-mers are used for building the GC bias curve. hg19_50_window.bed and hg19_500_window.bed are the window files in 50 or 500 30-mers per bin used for track displaying and smoothing. User can easily redefine the window in section 9. Finally, hg19_uniq.bc is the bloom counter for Jellyfish-2 which will speed up the QuicK-mer counting process and reduce I/O load.

8. Working Example

Here we use an example using public data from the NCBI short read archive to demonstrate QuicK-mer usage. Here we assume you are in your working directory.

1) **Download NA19240 sequencing file from SRA.**

wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR136/SRR1364052/SRR1364052.sra

fastq-dump -O SRR1364052 --split-files --gzip SRR1364052.sra

2) **Download hg19 30-mer reference**
We premade the 30-mer list for hg19 reference genome.

wget http://kiddlabshare.umms.med.umich.edu/public-data/QuicK-mer/Ref/hg19.tar.gz

tar xzfv hg19.tar.gz

3) **Running the QuicK-mer**
Add the following command to a job submission script and request 2 CPU cores with 35GB of total memory.

cd SRR1364052/

start_kmer_pipeline.py 19485_ATGTCA_L007*.fastq.gz –o 19485_ATGTCA_L007 hg19/

This process usually takes 6 hours. Once done, QuicK-mer will generate 3 files under the SRR1364052/ directory:

19485_ATGTCA_L007_result.bin
19485_ATGTCA_L007.txt
19485_ATGTCA_L007.PNG

The text file and PNG image record the GC-depth bias in the control region. The binary file 19485_ATGTCA_L007_result.bin contains all the GC-corrected depths for all 30-mers.

4) **Merge data**
To merge multiple GC-corrected depth files, move all the *_result.bin files into a directory and execute the following command from that directory:

```
ls *_result.bin > sample_name.txt
CorDepthCombine –l sample_name.txt
```

The result sample_name_merged.bin will contain the merged depth data for the files specified in sample_name.txt text file.

5) **Integrate browser track**
Finally, the user needs to convert the depth file into the bedGraph format based on predefined or user-defined windows.

```
kmer2window 19485_ATGTCA_L007_result.bin ../hg19/k30_hg19_CN2.bin
../hg19/hg19_500_window.bed > 19485_copy_number.bedGraph
```

The hg19_500_window.bed is a file specifying the genome location and number of k-mers in each window. The file 19485_ATGTCA_L007_result.bin can be substituted with the merged binary file sample_name_merged.bin when dealing with samples from multiple libraries.

This file can be further indexed and compressed into UCSC bigwig format and displayed using the UCSC genome browser.

9. Custom k-mer list

The user can define the k-mer list for any genome besides the premade ones listed in Step 6. The list of k-mers should have the following tab-delimited format:

```
chr1   10454   10484   chr1-10455    CTAACCCTAACCCTCGCGGTACCCTCAGCC
chr1   10455   10485   chr1-10456    CGGCTGAGGGTACCGCGAGGGTTAGGGTTA
chr1   10456   10486   chr1-10457    AACCCTAACCCTCGCGGTACCCTCAGCCGG
chr1   10457   10487   chr1-10458    ACCCTAACCCTCGCGGTACCCTCAGCCGGC
chr1   10458   10488   chr1-10459    CCCTAACCCTCGCGGTACCCTCAGCCGGCC
chr1   10459   10489   chr1-10460    CCTAACCCTCGCGGTACCCTCAGCCGGCCC
```

The first three columns define the genomic location of k-mer with the fifth column defines the k-mer sequence. The file must be in tab-delimited format and sorted based on genomic location.

10. Generate supporting files for custom k-mer list approach

Once a custom k-mer list is given, user could easily create the 3 essential axillary files using the built in command line tools. Below, we use the hg19 30-mer list as a starting point to create the axillary files.

1) Bloom Counter

The bloom counter double counts each k-mer in the list and then feeds it into the Jellyfish-2 for bloom counter generation. Essentially, this step marks predefined k-mers as "high frequency" during the actual counting process. This will reduce I/O when building the k-mer database.

```
cd hg19/
make-fasta-from-kmer.py hg19_kmer.bed | jellyfish-2 bc -C -m 30 -s 3G -t 16 -o kmer/ hg19_uniq.bc
/dev/fd/0
```

2) GC content

To generate GC content binary file, you'll need the reference genome files in FASTA format with sequence layout as 50bp per line.

```
cd hg19/
generate_GC_bin.py hg19_kmer.bed genomes/hg19/fasta/ k30_hg19_GC.bin
```

3) Window segments

Use the following command to make the window file for an existing k-mer list. The first argument "50" indicates 50 k-mers per window. The user can increase this value in order to trade finer resolution for minimization of the signal-to-noise ratio.

```
make_window_kmer.py 50 hg19_kmer.bed > hg19_50_window.bed
```