Canvas Software Design Document

Introduction

Canvas is an algorithm for calling copy number variants from either (a) a mostly diploid germline sample, or (b) a germline sample together with a tumor sample from the same individual. The vast majority of normal germline samples will be diploid, that is, having two copies. However, tumor samples may be much more extensively rearranged. Canvas seeks to identify regions of the sample's genome that are present in zero, one, or more than two times in the genome. Briefly, this is achieved by scanning the genome for regions that have an unexpected number of short read alignments. Regions with fewer than the expected number of alignments are classified as losses. Regions having more than the expected number of alignments are classified as gains.

Canvas was originally developed to work on low-depth sequencing data from cytogenetic, low-depth single-cell and whole genome sequencing experiments. Since then it has been extended to handle tumor/normal samples and targeted protocols such as whole exome tumor/normal data. Canvas can also be run on large custom targeted enrichment panels (e.g. a 100,000-probe panel such as exome data); however, using Canvas on targeted amplicon data (TruSeq Amplicon) is not appropriate.

The Canvas source code is distributed as an open-source software, and can be accessed at: https://github.com/Illumina/canvas

Canvas is described in the publication **Canvas: versatile and scalable detection of copy number variants** in the journal Bioinformatics.

> Publication link: http://bioinformatics.oxfordjournals.org/content/early/2016/04/19/bioinformatics.btw163
>
> BioRxiv link: http://biorxiv.org/content/early/2016/01/13/036194

Canvas supports a number of different workflows depending on the input sequencing data. The available modes are:

- **Germline-WGS:** CNV calling of a diploid germline sample from whole genome sequencing data
- **Somatic-Enrichment:** CNV calling of a somatic sample from targeted sequencing data
- **Somatic-WGS:** CNV calling of a somatic sample from whole genome sequencing data
- **Tumor-normal-enrichment:** CNV calling of a tumor/normal pair from targeted sequencing dat

Usage

Each of these workflows can be launched from the Canvas.exe command-line tool. An example of the usage information for the Somatic-WGS workflow is shown below.

```
[~]$/path/to/mono Canvas.exe Somatic-WGS -h
Canvas 1.10.0 Copyright © Illumina 2016
Somatic-WGS - CNV calling of a somatic sample from whole genome sequencing
data
Usage: Canvas.exe Somatic-WGS [OPTIONS]+
Options:
  -b, --bam=VALUE            tumor sample .bam file (required)
      --somatic-vcf=VALUE    .vcf file of somatic small variants found in
                               the tumor sample (required)
      --b-allele-vcf=VALUE   vcf containing SNV b-allele sites (only sites
                               with PASS in the filter column will be used)
                               (required)
      --exclude-non-het-b-allele-sites
                             exclude SNV b-allele sites from the vcf that
                               do not have sufficient read evidence for a
                               heterozygous genotype in the sample. This
                               option should be used when the b-allele
                               vcf is not matched to the sample
                               (e.g. from dbSNP)
      --ploidy-bed=VALUE     .bed file containing regions of known ploidy
                               in the sample. Copy number calls matching
                               the known ploidy in these regions will be
                               considered non-variant
  -o, --output=VALUE         output directory (required)
  -r, --reference=VALUE      Canvas-ready reference fasta file (required)
  -g, --genome-folder=VALUE  folder that contains both genome.fa and
                               GenomeSize.xml (required)
  -f, --filter-bed=VALUE     .bed file of regions to skip (required)
  -n, --sample-name=VALUE    sample name (required)
      --custom-parameters=VALUE
                             use custom parameter for command-line tool.
                               VALUE must contain the tool name followed
                               by a comma and then the custom parameters.
                               Option can be specified multiple times.
  -h, --help                 show this message and exit
  -v, --version              print version and exit
```

Below is an example script to launch the workflow using GRCh37 as the reference genome. It must be run on a dedicated compute node (e.g. through SGE using qsub).

```
#!/usr/bin/env bash
/path/to/mono Canvas.exe Somatic-WGS --bam
'/bioinfoSD/eroller/COLO829C/COLO829C_S1.bam' --sample-name
'COLO-829C-3lanes' --reference
'/illumina/sync/software/unofficial/Isas/Genomes/Homo_sapiens/Ensembl/GRCh
37/Annotation/Canvas/kmer.fa' --genome-folder
'/illumina/sync/software/unofficial/Isas/Genomes/Homo_sapiens/Ensembl/GRCh
37/Sequence/WholeGenomeFasta' --filter-bed
'/illumina/sync/software/unofficial/Isas/Genomes/Homo_sapiens/Ensembl/GRCh
37/Annotation/Canvas/filter13.bed' --output '/bioinfoSD/eroller/CanvasTest'
--b-allele-vcf '/bioinfoSD/eroller/COLO829BL/COLO829BL_S1.vcf.gz'
--somatic-vcf
'/bioinfoSD/eroller/COLO_smoke/COLO-829BL-2lanes_COLO-829C-3lanes_G1_P1.so
matic.vcf.gz' --ploidy-bed
'/bioinfoSD/eroller/COLO_smoke/DetectCNVTemp/COLO-829C-3lanes/ploidy.bed.g
z'
```

Canvas Overview

Outline The Canvas workflow comprises five distinct modules designed to (1) process aligned read data and calculate coverage bins, (2) perform outlier removal and normalization of coverage estimates, (3) identify segments of uniform copy number, (4) calculate minor allele frequencies (MAF) and (5) assign copy number / allelic states and infer genome-wide parameters. Separate workflows exist for somatic, germline and exome sequencing data. The latter workflow can be run either with or without a matched normal control sample and requires a manifest file with locations of targeted regions. For brevity, here we highlight core functionalities of individual modules,

Coverage binning Binning is performed by fixing a total number of unique k-mer alignments within each bin, leading to variable window size that depends on repeat content and alignment complexity of genomic regions.

Normalization Options for both bin-level and fragment-based GC-content normalization exist, along with a number of outlier removal strategies.

Partitioning and MAF calculation Canvas provides implementation of two coverage partitioning algorithms: unbalanced Haar wavelet transform (used by default) and circular binary segmentation (Fryzlewicz, 2007; Venkatraman et al., 2007). MAFs are derived at heterozygous SNV positions determined either by germline SNV variant calls in the reference (normal) sample or by direct assessment using population SNV panels (such as the 1000 Genomes Project or dbSNP database).

Somatic model and copy number (CN) assignment for tumour-normal whole-genome and exome workflows Accurate somatic copy number assignment requires knowledge of genome ploidy, sample contamination and location of heterogeneous variants. To infer these parameters Canvas fits a deviation model based on the principle that for a given combination of purity (normal contamination level) and diploid coverage level, MAF and coverage values for each CN state can be easily derived. Iterating over different purity and coverage combinations Canvas calculates a model deviation *D* by comparing observed and expected MAF and coverage values. To better discriminate models with similar deviation values, a separate penalty term *P* is introduced to measure the complexity of each model. The penalty is related to the total number of implied somatic events needed to produce a given tumor genome. Variables that best predict genome complexity are inferred using logistic regression on a training set of sequenced tumor genomes with known karyotypes. Finally, a separate EM clustering module is used to group *MAF, Coverage* tuples into clusters and assess if they could represent heterogeneous variants. This is done by computing distance between observed cluster centroids and expected *MAF, Coverage* values under the null (no heterogeneity) model. The resulting cluster deviation *C* is combined with other two metrics to provide a total deviation *T = D (model deviation) + P (penalty term) + C*. Ploidy, contamination and heterogeneity values from the model with the smallest total deviation are then used to assign CN and genotype to each segment.

CN assignment for germline variants Canvas can also detect germline CNVs from single-sample whole genome sequencing experiments. Since large scale changes in genome ploidy or sample contamination are rarely, if at all, present in "normal" samples, Canvas assigns copy numbers based on the premise that the majority of genomic regions are diploid and that there is no contamination (as seen with somatic genomes).

Below is a detailed description of individual modules.

Modules

Canvas workflows are composed of a number of distinct steps. Using the single executable described above will launch the appropriate steps using default parameters. The steps used in Canvas are:

(1) Counting alignments in genomic bins (implemented as CanvasBin.exe)

(2) Removal of systematic biases and outliers from the counts (implemented as CanvasClean.exe)

(3) Partitioning the counts into homogenous regions (implemented as CanvasPartition.exe)

(4) Quantifying SNV allele frequency for each region, including heterozygous-only SNVs identified in the normal sample for tumor/normal workflow (implemented as CanvasSNV.exe)

(5) Assigning a copy number to each homogenous region (implemented as CanvasSomaticCaller.exe or CanvasDiploidCaller.exe)

Below is a description of individual steps.

CanvasBin

The CanvasBin procedure creates genomic windows, or bins, across the genome and counts the number of observed alignments that fall into each bin. The alignments are provided in the form of a BAM file. Canvas has been validated using BAM files created using the Isaac and Bowtie aligners; results on .bam files from another aligner (e.g. bwa) would also likely be appropriate. The specific bowtie command appropriate for Canvas is of the form

```
zcat R1.fastq.gz | cut -b 1-35 | bowtie --quiet -S -q -v0 -m1 hg19 - | samtools view -Sbo FromBowtie.bam -
```

In this example, hg19 is a bowtie index generated for the hg19 genome built using bowtie's bowtie-build command. Note that only the first 35 bases of read 1 are aligned and stored in the BAM file. It follows that CNV-only applications such as low-depth cytogenetics require the sequencing of just 35 bases from one end of each sequencing template.

Isaac BAM files are created through calls made within the whole genome sequencing (WGS) workflow. Canvas applies special treatment to BAM files when measuring coverage. Specifically, Canvas will only keep alignments from proper pairs that align to the forward strand. Additionally, the alignment's CIGAR string must begin with at least thirty-five matches. It was found that restricting Isaac alignments in this way yielded very similar results to using Bowtie alignments of the same input data. To invoke this behavior at the command-line, the "-p" flag must be used when calling CanvasBin.exe.

CanvasBin stores observed alignments in RAM as a collection of byte arrays, one array for each chromosome. Each array has the same length as the the size of a corresponding chromosome. As the BAM file is read in, CanvasBin records the position of each alignment's left-most base within the chromosome-appropriate byte array. After all alignments in the BAM file have been read, the byte arrays have a nonzero integer wherever an alignment was observed and a zero everywhere else. The -m (coverage mode) argument to CanvasBin controls the details of how these counts are accumulated; the default mode "TruncatedDynamicRange" counts up to 10 hits at each position.

In addition to a BAM file, CanvasBin accepts FASTA **kmer.fa** file as input (provided in the Annotation/Canvas subfolder for a reference genome) that contains genomic sequences that were used for alignment. The first base of each 35-mer within this FASTA file is marked as unique or non-unique with uppercase and lowercase letters. If a 35-mer is unique then its first nucleotide is capitalized, otherwise, it is not capitalized. Note: In previous workflow versions (2.6.6 and earlier), the kmer file was named **genome.fa**; the new name is clearer and includes correct handling of PAR-masked regions.

For example, in the sequence

acgtttaATgacgatGaacgatcagctaagaatacgacaatatcagacaa

, the three 35-mers marked as unique would be

ATGACGATGAACGATCAGCTAAGAATACGACAATA

TGACGATGAACGATCAGCTAAGAATACGACAATAT

GAACGATCAGCTAAGAATACGACAATATCAGACAA

The genomic locations of unique 35-mers are stored by CanvasBin in a collection of bit arrays analogous to those used to store alignment positions. Unique positions and non-unique positions are marked with "1"s and "0"s, respectively. The tool **FlagUniqueKmers**, from the Canvas solution, can be used to generate this uniqueness-flagged annotation for a reference FASTA file.

If a Bowtie BAM is used, then the alignment byte arrays will never have a nonzero value in a position where the corresponding position in the unique 35-mer byte array is a "0". This is because bowtie, if used in the appropriate way for Canvas (see example earlier in this section), will only return exact 35-mer matches that are unique in the genome. Isaac is a more sensitive aligner and can place reads in locations that are not prefixed by a unique 35-mer. Therefore, CanvasBin next iterates through the byte array for hit counts, and sets this to "0" everywhere that the unique 35-mer byte array is set to "0". This guarantees that we only use Isaac alignments that start at unique 35-mer positions in the genome.

Excluded Regions

Optionally, a subset of unique 35-mers can be masked from analysis. A BED file can be provided at the command line that specifies regions of the genome to ignore from analysis. After the alignments and unique 35-mers are loaded, the regions in this BED file are read in. For each entry in the file, the elements of both the alignment and unique 35-mer byte arrays that are contained within the entry's range are set to "0". This effectively excludes the region from further analysis and is therefore a means to ignore problematic regions of the genome. A default masking file

is provided with Isis/Canvas (now named filter13.bed - was named filter.bed in previous releases) which includes such regions. It was generated by sequencing a pool of 100 normal individuals and calling CNVs with Canvas. Since the “sample” is really a pool of many samples, any detected region is presumably an artifact of the reference genome or the secondary analysis pipeline - e.g. telomeres and centromeres (and some pericentromeric regions) are excluded.

Bin Size

When CanvasBin is invoked, the user supplies a desired average number of alignments per bin for diploid regions. By default this value is set to 100 alignments per bin for WGS data. (This default was chosen to provide roughly the same signal-to-noise ratio as an Agilent CGH array.) For enrichment data, the default is set to 300. This value must then be converted into a “bin size.” In typical usage, “bin size” would refer to the size of some fixed genomic window. An example of this might be a bin size of 10kb. Here, we use the term “bin size“ to refer to the number of unique genomic 35-mers per bin. Because some regions of the human genome are more repetitive than others, bins’ physical sizes (in genomic coordinates) will not be identical. Below is a toy example to illustrate this concept. Each box is a position along the genome. Each checkmark represents a unique 35-mer while each X represents a non-unique 35-mer. The bin size here is three (three check marks per bin). Note that the physical size of each bin is not constant. B_1 and B_3 have a physical size of three but B_2 and B_4 have physical sizes of four and six, respectively.

To compute bin size, the ratio of observed alignments to unique 35-mers is calculated for each autosome. The desired number of alignments per bin is then divided by the median of these ratios to yield bin size. For whole genome sequencing, bin sizes will typically be in the range of 800-1000 unique 35-mers. Correspondingly, the majority of physical window sizes will be in the 1-1.2kb range. The advantage of this approach relative to using fixed genomic intervals for bins is that we now expect to have the same number of reads mapping to each bin, regardless of the bins’ “mappability” or “uniqueness”. Optionally, CanvasBin takes a Nextera manifest file through command line argument -t <path to manifest> (–manifest=<path to manifest>). In this case, only the targeted regions specified in the manifest are used to compute the bin size such that bins overlapping the targeted regions (the on-target bins) have the desired median number of alignments.

Bins

Once bin size is computed, bins are defined as consecutive genomic windows such that each bin contains the same number, bin size, of unique 35-mers. The number of observed alignments present within each bin’s boundaries is then counted from the alignment byte arrays. The GC content of each bin is also calculated.

CanvasBin supports user-defined bins specified by -n <path to a BED file>. This option is particularly useful for enrichment data. Probe regions and exons are commonly used for binning. Since user-defined bins are not guaranteed to contain the same number of unique 35-mers per bin, downstream coverage normalization is needed such that each bin follows the same distribution.

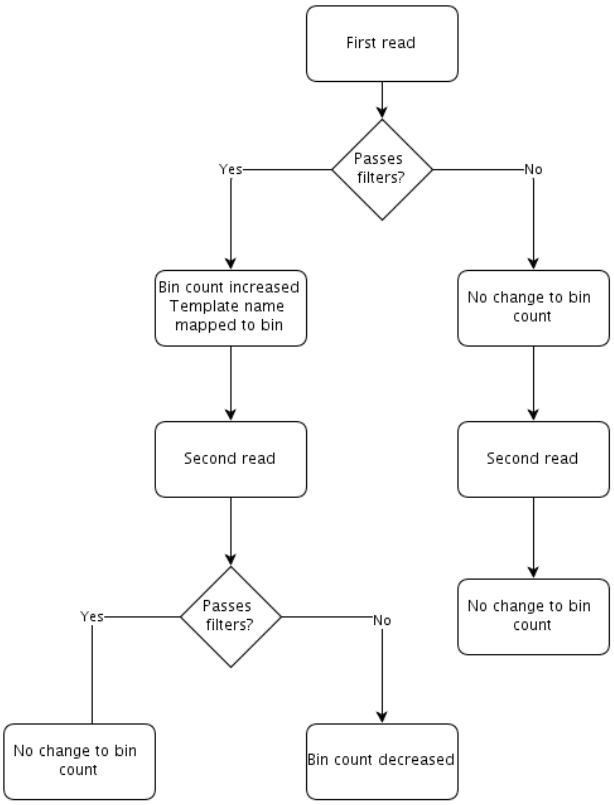
Fragment Binning

CanvasBin supports fragment binning with user-defined bins. The fragment binning mode is turned on by command line argument -m Fragment. The algorithm works by assigning sequenced fragments to bins. The aim is to associate each bin with a number of fragments and report these numbers as the counts for the bins. The start and end positions of fragments are calculated from the locations of paired-end reads in the input BAM file. A fragment will only contribute towards the count for a particular bin if both reads in the pair pass a set of filters. Each fragment is assigned to a bin according to which bin the fragment overlaps the most. If a fragment overlaps more than one bin by the same amount, it will be assigned to the left-most (smallest start position) bin.

Each read is subject to a number of quality filters before it can form part of a valid fragment. These filters are split into three categories:

1. Are the read and its mate properly paired?
2. Is the read a unique?
3. Is the read a high-quality read with a good alignment?

The algorithm operates in such a way that only fragments where both reads in the pair pass the above filters contribute to a bin's count and it requires one pass through the BAM file. To do this, the first and second reads are considered separately. The fragment start and end can be identified from each individual alignment record. If the first read in a pair passes the filters, the count for the corresponding bin will be incremented and a unique name associated with this pair of reads will be mapped to the bin. When the second read in the pair is encountered in the pair, if it doesn't pass the filters and the first read has been used to increment a count, the bin's count will be decreased by one. The following flowchart ou tlines the changes to the bin counts as the first and second reads are encountered in the BAM file.



The following table shows the changes to the bin count that will be made as the first and second read in a pair are encountered. The result of this algorithm is an integer count associated with each bin.

First Read	Bin Count Change	Second Read	Bin Count Change	Overall Change
Passes filters	+1	Passes filters	+0	+1
Passes filters	+1	Fails filters	-1	+0
Fails filters	+0	Passes filters	+0	+0
Fails filters	+0	Fails filters	+0	+0

Output Format

The output of CanvasBin is a **G1_P1_0.binned** gzip bed file containing chromosome, genomic start, genomic stop, observed counts and GC content for each bin. For illustration, the first five lines of one such file is shown below:

```
chr1    754148  755200  90      46
chr1    755200  756030  93      54
chr1    756030  756783  90      54
chr1    756783  757489  89      53
chr1    757489  758242  80      55
```

CanvasClean

CanvasClean is comprised of four procedures for removing outliers and systematic biases from the count data computed in CanvasBin. These are

(1) Single point outlier removal

(2) Physical size outlier removal

(3) GC content correction

(4) Normalization of formalin-fixed, paraffin-embedded (FFPE) samples (somatic workflows only)

CanvasClean takes as input the BED file created by CanvasBin.

Single Point Outlier Removal

The aim of this step is to remove individual bins that represent extreme outliers. These are bins whose counts are very different from the counts present in the bins upstream and downstream from themselves. We define two values, *a* and *b*, to be very different if their difference is greater than we would expect by chance, assuming *a* and *b* come from the same underlying distribution. Specifically, we make use of the Chi-squared distribution:

$\mu = 0.5a + 0.5b$

$^2 = ((a - \mu)^2 + (b - \mu)^2)\ \mu^{-1}$

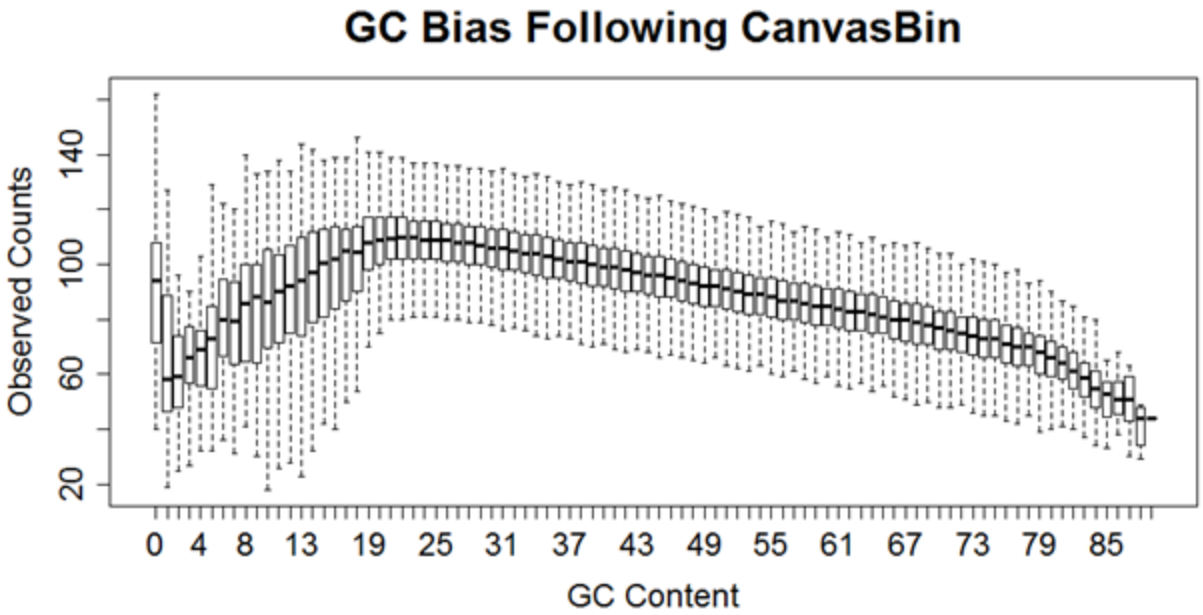
A value of ^{2} greater than 6.635, which is the 99th percentile of the Chi-squared distribution with one degree of freedom, is considered “very different.” If a bin’s count is very different from both its immediate upstream and downstream neighbors’ counts, then the bin is deemed an outlier and removed.

Physical Size Outlier Removal

Recall that bins will likely not have the same physical (genomic) size. The average for whole genome sequencing runs might be approximately 1kb but some may be several megabases in size if they cover repetitive regions of the genome. Example regions might include centromeres and telomeres. The counts in these regions tend to be unreliable so we remove bins with extreme physical size. Specifically, we calculate the 98th per centile of observed physical sizes and remove bins whose sizes are larger than this threshold.

GC Content Correction

The main driver of variability in bins’ counts is their GC contents. CanvasClean provides a mechanism for correcting any GC content bias that may be present in the counts calculated in CanvasBin. An example of the magnitude of the bias is presented in the following figure.



Two GC normalization modes are available, MedianByGC and LOESS.

MedianByGC

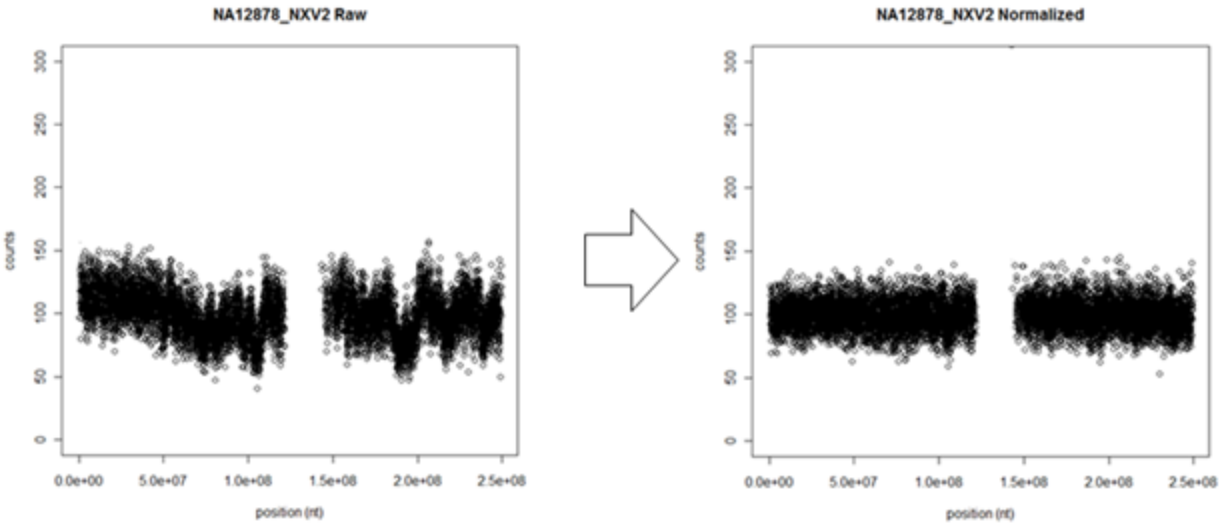
The correction is simple. Bins are first aggregated by their GC content, which has been rounded to the nearest integer. Second, each bin’s count is divided by the median count of bins having the same GC content. Finally, this value is multiplied by the global median. The effect is to flatten the midpoints of the bars in the box-and-whisker plot displayed above.

Some values for GC content will have very few bins so the estimate of its median is not very robust. To guard against this, we discard any bin where the number of bins sharing the same GC content is fewer than a threshold computed by max(minNumberOfBinsPerGCForWeightedMeidan, number of bins/100). If a GC value has fewer than 100 bins, the weighted median is

computed using bins of neighboring GC values. Bins of the target GC value get full weight, bins of the two neighboring GC values get half weight, two-away bins get 1/4 weight, etc. Neighboring GC values are included until we have at least 100 bins to estimate the weighted median bin count. Parameter minNumberOfBinsPerGCForWeightedMeidan can be set using command line argument -w <number of bins>.

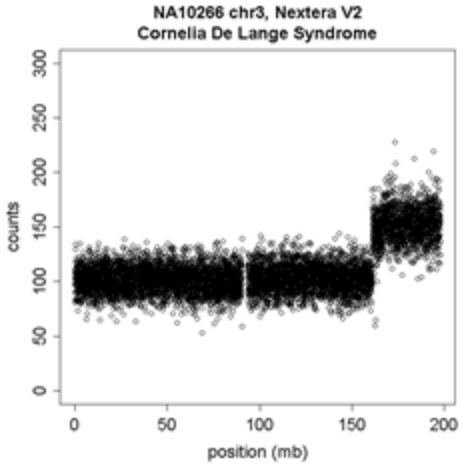
Optionally, CanvasClean takes a Nextera manifest file through command line argument -t <path to manifest> (–manifest=<path to manifest>). In this case, only on-target bins are used to estimate the count distribution at each GC content level. Similarly, we discard any any bin where the number of on-target bins sharing the same GC content is fewer than a threshold computed by max(minNumberOfBinsPerGCForWeightedMeidan, number of on-target bins/100). The weighted median is similarly computed when there are less than 100 bins for a GC value.

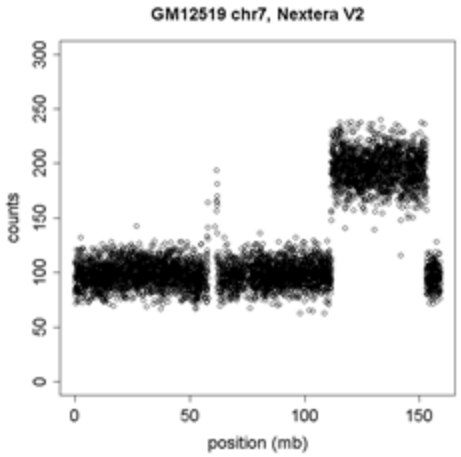
For some sample preparation schemes, GC content correction has a dramatic effect. The figure below illustrates this for a low depth sequencing experiment using the Nextera library preparation method. The figure on the left shows bins counts as a function of chromosome position before normalization. The figure on the right shows the same, following GC content correction.



For whole genome sequencing experiments, we typically see median absolute deviations (MADs) of 10.3, which is very close to the expected value of 10 predicted by the Poisson model for an average count of 100, indicating that little bias remains following the simple procedures implemented in CanvasClean.

Importantly, the normalization signal does not dampen signal from CNVs. Below are two figures, one for a chromosome known to harbor a single copy gain and one for a double copy gain.





LOESS

This GC normalization mode can be turned on by -m LOESS or --mode LOESS.

In the LEOSS mode, GC normalization works the same as in the MedianByGC mode except that the fitted bin count for each GC percentage is obtained from fitting a LOESS model. The bin counts are first log-transformed for variance stabilization, normalized by GC content, and transformed back after normalization.

The degree of the polynomials used in LOESS is 1. The bandwidth is searched in [0.3, 0.75] to find the value that minimize an objective function **without using chromosome Y**. Given a bandwidth, GC percentages and bin counts, the fitted bin counts are obtained by LOESS to calculate the normalized bin counts:

"normalized bin count" = "bin count" - "fitted bin count" + "median bin count".

Another LOESS is performed on the normalized bin counts and the standard deviation of the fitted values is calculated. This is because, after two passes of LOESS, the standard deviation is expected to be small when there is no CNV.

Once the best bandwidth is found, a LOESS model is fitted using **all the chromosomes**. The normalized bin counts are calculated and exp-transformed.

Normalization of FFPE samples

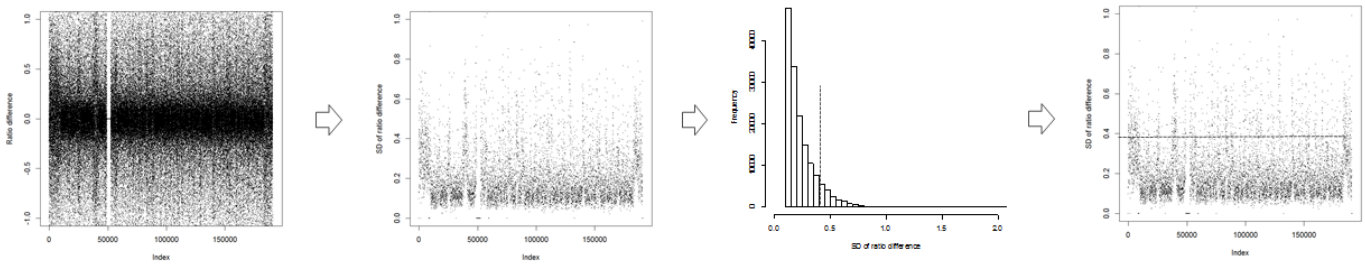
FFPE procedure is a common way of preserving tumor tissues following a cancer treatment surgery and is therefore a frequent source of material for tumor sequencing. However, FFPE fixation introduces noise and biases during sample preparation (fragmentation, need of high temperatures for DNA extraction) that lead to difficult-to-interpret coverage depth anomalies. While some of it is traced to the dropout of AT-rich sequencing fragment (in part corrected by GC-normalization), it doesn't fully explain the biases suggesting that other poorly understood factors, like chromatin confirmation and 2/3D DNA interaction might be involved. Canvas uses a two-step strategy to normalize FFPE samples:

(1) FFPE de-noising algorithm to remove localized biases

(2) Fragment-based normalization

FFPE de-noising algorithm

Canvas uses a separate FFPE de-noising algorithm that utilizes FFPE-specific properties of noise and variance. In particular, the main feature of FFPE-induced noise is an increase in coverage variance in regions of high bias, which could not be corrected by normalization routings relying on mean and related summary statistics (median, mode, etc.). Specifically, the method is composed of the following steps (also show on the pictogram):



Derive time-lag (=1) coverage ratio difference (**Skellam Distribution:** Mean = 0; SD= 2*original Data SD)

Estimate standard deviation (SD) of coverage differences in a window (default 20 bins)

Compare statistics (MAD, median) of the distribution on large number of FFPE and FF samples to derive threshold

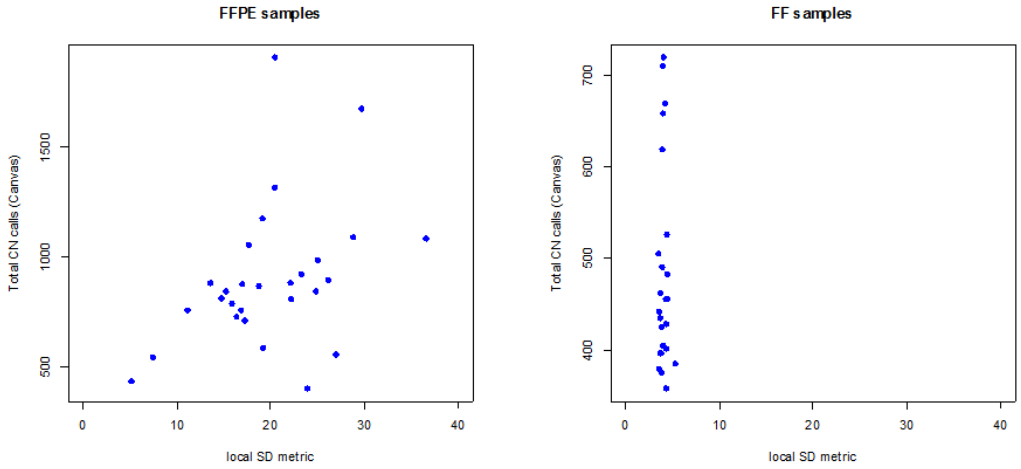
Remove outliers above threshold

1) Estimate time-lag difference of consecutive coverage bins. Time-lagged difference reveals CN-independent spikes of variance in FFPE-bias regions. The advantageous properties of the transformation include:

1. The mean for all regions (including large scale CN changes) is zero;
2. The variance is doubled (http://en.wikipedia.org/wiki/Skellam_distribution) - making regions of noise easier to detect
3. Correct breakpoint regions can be distinguished by sharper transition between CN changes than the regions of choppiness.

2) Such spikes can be used to implement a simple local FFPE bias/variance de-noising algorithm. Canvas runs a sliding window of coverage bins (20 bins by default) and calculates standard deviation in each window. The average (SD) over all windows and chromosomes is then estimated for each sample; for simplicity such a metric is called local standard deviation (localSD). Bins with localSD threshold are removed at a CanvasClean stage. Since this is a lossy transformation, the threshold is set such that only noisy FFPE samples are de-noised, while normal and less noisy FFPE samples are not affected (step 3 below for details).

3) Comparing localSDs of coverage data from a reference set of FFPE and FF samples derived from the same individuals clearly separates FFPE and FF groups by the metric (**localSD threshold set to 10** from examples below). This property is used to set a localSD threshold specifically to insure that only noisy FFPE samples undergo de-nosing.



Fragment-based normalization

While bin-level normalization corrects most of the noise, it still does not capture variability that occurs at the level of individual fragment sizes. The later is particularly influenced by PCR amplification step occurring during FFPE library preparation workflow. To improve it, Canvas normalizes bin counts through a weighted GC-content correction factor, derived by averaging contribution of individual fragments towards the total bin counts. A fragment average is weighted by ratio of observed versus expected GC-content counts for a given fragment sequence.

CanvasClean writes to disk a file of the same format as was written by CanvasBin.

CanvasPartition

CanvasPartition takes as input normalized CanvasClean BED files and segments all bins into set of distinct segments having equal mean coverage values. CanvasPartition provides implementation of a number of segmentation algorithms: currently these include unbalanced Haar wavelet segmentation (the default) and circular binary segmentation (CBS).

Haar wavelet segmentation is an alternative partitioning method introduced in (Fryzlewicz P, 2007). It is based on a traditional discrete wavelet transform (DWT), but in contrast to DWT allows for an arbitrary size segments (rather then size factor of two). This is achieved by using adaptation of matching pursuit algorithm (see paper for details) that allows breakpoint basis selection in O(Nlog(N)) time. The resulting DWT coefficients along with original Haar basis vectors can be used to recontact a de-noised signal after setting all DWT coefficients below user specified threshold to zero. This smoothed piecewise constant signal can be used to characterize breakpoints in CanvasClean output by looking at a difference between consecutive data points.

CBS is an algorithm for identifying regions of the genome such that their average counts are statistically different than their neighboring regions' average counts. The implementation is a port of the circular binary segmentation (CBS) algorithm from its original C/Fortran to C#. The algorithm is fully described in (Olshen AB, 2004) and (Venkatraman ES, 2007). Briefly, each chromosome is considered as a segment. The algorithm looks into each chromosome/segment and identifies the pair of bins for which the counts in the bins between them are maximally different than the rest of the bins' counts. The statistical significance of the maximal difference is assessed via permutation testing. If the difference is found to be statistically significant, then the procedure is applied recursively to the two or three segments created by partitioning the current segment by the identified pair of points. Input to the algorithm is the output generated by the CanvasClean algorithm. The computational complexity of the algorithm $O(N^2)$ so the problem is divided into sub-chromosome problems followed by merging, in practice. Heuristics are used to speed up the permutation testing. The reader is directed to the referenced articles for a detailed discussion.

CanvasPartition outputs a gzipped-BED-like file (**G1_P1.partitioned**) containing chromosome, genomic start, genomic stop, bin coverage and an integer label that identifies which partitioned region the bin belongs to. For illustration, the first five lines of one such file are shown below:

```
chr1    754148  755200  90      1

chr1    755200  756030  93      1

chr1    756030  756783  90      1

chr1    756783  757489  89      1

chr1    757489  758242  80      1
```

CanvasSNV

CanvasSNV step is used to estimate SNV allele frequencies (aka minor allele frequencies, MAF). It takes as input (1) the variant calls (.vcf file) for the normal/reference sample or path to a dbSNP / reference vcf file, and (2) the aligned reads (.bam file) for the tumor/reference sample.

For the tumor/normal workflow, germline SNV calls from the normal sample are used to identify all heterozygous SNVs (GT = 0/1 or 1/0) that pass filters (PASS in the FILTER column). We also require that the GQX tag (if present) is >= 30. For germline calling, CanvasSNV can use either the germline SNV calls, or has the option to review all dbSNP sites - see Resequencing Workflow Software Design Document)

For each SNV site, CanvasSNV computes how many observations were made for the REF and ALT alleles. This calculation considers all reads which are not flagged as PCR duplicates, secondary or supplementary alignments. Next counts (for all reads whose alignment to the reference covers a site of interest) of how many reads have the reference base, and how many reads have the alternative base are made.

Given A and B reads for the two alleles being considered (where A+B>0), we define the minor allele frequency (MAF) as min(A, B) / (A + B). Note that MAF will be approximately 0.5 for diploid regions. (In practice, the MAF at diploid positions will be a bit less - the higher the coverage, the closer to 0.5). For regions with a given **ploidy** (copy number and major allele count), we can predict the minor allele frequency of SNVs in the region:

- Ploidy A: MAF 0
- Ploidy AB (normal): MAF 0.5
- Ploidy AA (copy-neutral LOH): MAF 0.5
- Ploidy AAB: MAF 0.33333
- Ploidy AAA: MAF 0
- Ploidy AABB: MAF 0.5
- Ploidy AAAB: MAF 0.25
- Ploidy AAAA: MAF 0
- (etc.)

The output of CanvasSNV is a temporary file **VFResultsG1_P1.txt.gz**. It is a gzipped tab-delimited file containing one header line (beginning with #), an where each record contains the following fields: chromosome name, position, reference allele, alternate allele, # of reads for the reference allele, # of reads for the alternate allele. Here are few example lines from such as file:

```
#Chromosome Position Ref Alt CountRef CountAlt
chr1 12783 G A 53 135
chr1 13116 T G 24 70
chr1 13118 A G 25 69
chr1 14907 A G 73 102
chr1 14930 A G 48 98
chr1 15190 G A 160 13
chr1 16298 C T 18 54
```

CanvasDiploidCaller

The final step of the Canvas workflow for calling germline CNV variants is to assign discrete copy number states to each region identified by CanvasPartition and to determine variant's major copy number (MCC, copy number of major allele) where possible. CanvasDiploidCaller takes coverage and MAF values for each CanvasPartition segments as input. It uses Gaussian Mixture model with a separate component for each copy number and MCC states: e.g. for CN=2 model has two states representing diploid case (MCC=1, CN=2) and LOH (MCC=2, CN=2). Means and covariance matrices are estimated from the data for the diploid model and then adjusted via Expectation Maximization (EM) algorithm. For example, if the mean, μ, and standard deviation, , are estimated to be 100 and 15 in the diploid model, then the corresponding estimates in the haploid model would be initialized to μ/2 and /2. Following EM fitting, [CN,MCC] model tuple with the highest posterior probability is used to assign copy number to each segment. In cases where a segment has no spanning SNPs and hence empty MAF values, LOH states are skipped and not considered.

Following assignment of copy number states, neighboring regions that received the same copy number call are merged into a single region. This can occur if CanvasPartition over-segments a region.

Then Phred-like *q*-scores are assigned to each region. A model has been trained on a test corpus of ~10 curated germline samples to compute the probability that each segment has the correct direction - i.e. true copy number is 2 and it was assigned copy number 2, true copy number is <2 and it was assigned copy number <2, or true copy number is >2 and it was assigned copy number >2. A logistic regression model was trained to compute this probability. The three features used are:

- *LogBinCount* $\log_{10}$(1+number of bins). Segments with more bins can be assigned a copy number more confidently
- *ModelDistance*: Distance to the model. (Lower distances reflect a better fit)
- *DistanceRatio:* Ratio between between the distance to the best model point, and the distance to the next runner-up. The larger this value, the greater the confidence that the assigned ploidy is indeed correct.

The probability is then scaled to generate a Phred-scaled likelihood that the call is correct. This estimate is likely to be conservative as it is derived from array CGH data which is likely not 100% accurate. Importantly, *q*-scores are a function of number of bins, not genomic size, so they are applicable to experiments of any sequencing depth, including low-depth cytogenetics screening.

The coordinates of non-diploid regions and their *q*-scores are output to a VCF file. Two filters are applied to PASS variants. First, a variant must be have a *q*-score of Q10 or greater. Second, a variant must be of size 10kb or greater.

CanvasSomaticCaller

CanvasSomaticCaller is used in tumor-normal and somatic workflows to identify regions with an abnormal copy number (e.g. for autosomes the copy number != 2 or b-allele frequency != ~50%). It does this by accounting for any changes in ploidy, normal contamination (purity) and polyclonality. The correction logic for these confounding factors is implemented in somatic model, key principles of which are:

Principle 1: For a given set of ploidy and purity values one can derive **expected** values of coverage and MAF **for each copy number state.**

Definition: Model Deviation = (expected – observed) coverage and MAF values for a given set of ploidy and purity values.

Principle 2: For two models with similar deviations, the model that implies a smaller number of CN changes (i.e. closer to diploid) should be preferred.

Principle 3: Given training samples with known ploidy/purity values, optimization techniques can be used to infer the best feature weights to use for fitting a particular somatic model.

Principle 4: Given most likely purity and ploidy values, for a given segment, departure from **observed coverage and MAF values** can be indicative of segment's polyclonality

As mentioned, CanvasSomaticCaller attempts to estimate overall variant heterogeneity if the tumor is polyclonal. While finding ploidy/purity prediction is linear in the number of states, including heterogeneity makes model complexity exponential. CanvasSomaticCaller uses a number of approximations (described below) to detect polyclonality.

CanvasSomaticCaller is composed of a number of steps.

Input and pre-processing

1. Load coverage segments identified by CanvasPartition
2. Load variant frequency information captured by CanvasSNV; retain all sites with ref+alt coverage >= 10. CanvasSNV output contains only variant frequencies of heterozygous SNVs if the VCF of a matched normal sample was used. However, a dbSNP VCF can be used in place of a normal VCF and hence not all positions in CanvasSNV are heterozygous. In this case, the command line argument -d (--dbsnpvcf) should be used to prune MAFs of homozygous positions for each segment.
    a. We filter the zero MAFs for a segment if more than 10% of MAFs are non-zero or at least one of the positions has a MAF >= 0.2. This is because homozygous positions are expected to have MAFs of zero.
    b. When filtering is done, we further take only MAFs > 0.1 if there are any MAFs > 0.1 in the segment. This is because some homozygous positions have non-zero MAFs that are very close to zero.
    c. If the number of MAFs before filtering is greater than the given MAF count threshold (default: 50) but less than this threshold after filtering, we lower the threshold to be the number of MAFs after filtering.
3. For each segment identified by CanvasPartition, note the median number of counts per bin ("coverage") and the median minor allele frequency (MAF).
    a. The number of MAFs of each segment must meet the MAF count threshold (default: 50) to be used in model fitting. However, the MAF count threshold may be relaxed so that we have at least 20 segments for model fitting.
    b. Each segment is assigned a weight, which is used in computing the deviation described below. The weight is the length of the segment if the total number of segments is greater than 100. Otherwise, the weight is the number of bins in the segment. In case we have fewer than 10 MAFs in the segment, the weight is multiplied by the ratio of the number of MAFs over 10. This is because a smaller number of MAFs will provide a less reliable estimation of the true median MAF for the segment. This weighting technique makes model fitting more robust to noise.

Somatic model fitting

1. Model the median diploid coverage **D** and the overall purity **p**: Given a candidate tuple **(D, p)**, compute the expected (MAF, Coverage) points for pre-specified set of ploidy and purity values. Loop over a collection of all segments of reasonable size (>50kb) and informative numbers of SNVs (>=50). The **model deviation** for (D, p) is the sum, across all segments, of ||(SegmentMAF, SegmentCoverage) - (ExpectedMAF, ExpectedCoverage)||. The final **total deviation** is derided by combining **model deviation** and **penalty term** that includes additional coefficient-weighted factors as follows:
   +*W1**DeviationScore: Model deviation
   +*W2**DiploidDistance: Scaled number of CN events needed to transform a given genome into diploid
   +*W3**CN2: Percentage of CN = 2
   +*W4**PercentNormal: Percentage of normal CNs
   To derive the formula coefficients *W1, W2, W3* and *W4*, a model training workflow used ~140 annotated tumour samples was used as described in https://git.illumina.com/Bioinformatics/CNAT/tree/master/CanvasSomaticTrainingData.
2. The model with the lowest score is then selected as a **preliminary best model of overall coverage and purity.**
3. Preliminary calling: For each segment, identify the ploidy which best fits the MAF and coverage of that segment, according to our model of overall coverage of purity.
4. A **merge** step is carried out. Adjacent segments with same copy number are combined into a single segment, and segments with size under a threshold (currently 50000 bases) are merged into the best (highest q-score) adjacent segment. Segments are considered adjacent if they are on the same chromosome, have no other segments between them, and have no forbidden intervals (from the filtering .bed file) between them)

Detection of polyclonal variants

1. Somatic caller also attempts to account for **tumor polyclonality** by implementing Principle 4 described above. The assumption used by CanvasSomaticCaller is that observed coverage and MAF values from polyclonal regions will tend to group into a limited set of clusters. Each segment tuple (SegmentMAF, SegmentCoverage) is then assigned to these distinct clusters using non-parametric density clustering (Rodriguez A., 2014) by default (--custom-parameters=CanvasSomaticCaller ,CanvasSomaticClusteringMode=Density) or Gaussian Mixture Expectation-Maximization clustering (--custom-parameters=CanvasSomaticCaller ,CanvasSomaticClusteringMode=GaussianMixture). **Cluster deviation** for cluster (D, p) is defined as the sum of distances between segments belonging to the cluster and the closest expected (MAF, Coverage) tuple for a given ploidy and purity model as calculated above. In general, clusters located farthest from expected (MAF, Coverage) centroid are more likely to be heterogeneous**. The adjusted total deviation is than defined as the average of total deviation and** cluster deviation.
2. Best somatic purity and ploidy model found in step 1 is used to identify polyclonal variants and correct CN assignment if necessary (from CN=2 to CN=1 or CN=3). The correction is done by finding the variants that satisfy following requirements:
    a. most likely CN=2
    b. second most likely CN=1 or 3
    c. best model purity > 0.2
    d. variant segment is assigned to heterogeneous clusters

    e. ratio of segment distance to the best and runner-up model is above threshold W5
3. Assign Phred-like *q*-scores to each region. A model has been trained on a test corpus of 33 curated tumor/normal samples to compute the probability that each segment has the correct direction - i.e. true copy number is 2 and it was assigned copy number 2, true copy number is <2 and it was assigned copy number <2, or true copy number is >2 and it was assigned copy number >2. A logistic regression model was trained to compute this probability. The three features used are:
    a. *LogBinCount* $\log_{10}$(1+number of bins). Segments with more bins can be assigned a copy number more confidently
    b. *ModelDistance*: Distance to the model. (Lower distances reflect a better fit)
    c. *DistanceRatio:* Ratio between between the distance to the best model point, and the distance to the next runner-up. The larger this value, the greater the confidence that the assigned ploidy is indeed correct.

    The probability is then scaled to generate a Phred-scaled likelihood that the call is correct. Note that this q-scoring model uses the same features as the model for germline calling (CanvasDiploidCaller), but re-trained on tumor/normal data. The overall ROC curve area on the test data was 0.829. (Note that truth data is not available, so some of the "false positives" may reflect correct copy number calls which have not been annotated correctly in the truth data. Ten of the highest-scoring "bad" calls on the training data were assessed to ensure they are plausible copy number calls)

Estimate purity from somatic SNVs

- If somatic SNV calls are available (typically generated by Strelka as part of the tumor/normal workflow), then the tumor purity is also estimated from these SNV calls. We take all SNV calls which pass filters and which have a variant allele frequency <0.5; the estimated tumor purity is twice the mean variant allele frequency of these somatic SNVs. This estimate is written to the Canvas log file. If we did not assign many CNVs (if 95% or more of the genome is at copy number 2), then this SNV-estimated purity is what gets reported to the output .vcf file. In tests across several data-sets (HCC1187 admixture data, Gel35, Gel9), the CNV-derived and SNV-derived tumor purity estimates agreed within 3%.

The main output of CanvasSomaticCaller is a variant call file (.vcf 4.1 format) listing all intervals which were assigned a copy number. See CanvasSomaticCaller outputs, below.

Copy numbers are reported as a REF, LOSS, or GAIN depending on the copy number count and expected count. The expected count is generally 2 (copy number 2 is REF, copy number 0 or 1 is LOSS, copy number 3 or greater is GAIN). The exception is that if a ploidy .bed file is passed in to Canvas, it can update the expected number of copies for particular chromosomes. This feature is used in the analysis workflow, based on the normal allosome (sex chromosome) copy count reported in the SNP/Indel .vcf header line ##ExpectedSexChromosomeKaryotype. For instance, for chrX and chrY on an XY sample, copy number calls of 2 on X or Y would be called as a GAIN, with the exception of the PAR region on chrX which still has expected copy number 2.

Parameter training

To facilitate data-driven parameter optimization, all parameters related to model fitting of CanvasSomaticCaller are stored in **SomaticCallerParameters.json** configuration file (under CanvasSomaticCaller subproject). optimizeSomaticCanvasModel.py training workflow takes this configuration file and a set of training samples as an input and attempts to optimize parameters given the data. The training data andscript reside under https://git.illumina.com/Bioinformatics/CNAT repository. The training procedure is fully described in https://git.illumina.com/Bioinformatics/CNAT/blob/master/README.md

Module operations

As exemplified in the previous sections, Canvas comprises different executable components wrapped into easy-to-use workflows for different sample types and enrichment / whole genome cases. While they can be invoked through a single command line to generate a set of variant call without any further interventions, users can also run individual components to generate intermediate files or to rerun a particular module with different parameter settings.

Below is an example command for CanvasBin

CanvasBin

```
mono CanvasBin.exe

>>>Command-line arguments:

CanvasBin [version number]

Please specify the Canvas k-uniqueness reference file.

Usage: CanvasBin.exe [OPTIONS]+

Bin alignments into variable-sized genomic intervals.

Options:

  -b, --bam=VALUE            bam file containing unique alignments
```

```
  -r, --reference=VALUE      Canvas-ready reference fasta file
  -c, --chr=VALUE            for bam input, only work on this chromosome.
                               Output intermediate binary data. Must follow-up
                               with a single CanvasBin call passing all the
                               intermediate binary data files (see -i option)
  -i, --infile=VALUE         intermediate binary data file from individual
                               chromosome. Pass this option multiple times,
                               once for each chromosome
  -f, --filter=VALUE         bed file containing regions to ignore
  -d, --bindepth=VALUE       median counts desired in each bin
  -z, --binsize=VALUE        bin size; optional
  -o, --outfile=VALUE        text file to output containing computed bins, or
                               if -c option was specified the intermediate
                               binary data file to output
  -y, --binsizeonly          calcualte bin size and exit
  -h, --help                 show this message and exit
  -p, --paired-end           input .bam is a paired-end alignment (e.g. from
                               Isaac)
  -m, --mode=VALUE           coverage measurement mode
  -t, --manifest=VALUE       Nextera manifest file
  -n, --bins=VALUE           bed file containing predefined bins
```

Default parameters at the individual modules step can be passed on to the main executable by invoking --custom-parameters=[module_name],-m=[parameter_name]. For example, to enable fragment-based GC content normalization beneficial for processing FFPE samples, CanvasBin parameter “mode” can be altered through command line argument of the main Canvas executable as follows: --custom-parameters=CanvasBin,-m=TruncatedDynamicRange.

In addition to running individual Canvas modules, Canvas binary allows running a subset of modules by utilizing checkpoints via -c and –s arguments:

```
  -c=VALUE                   continue analysis starting at the specified checkpoint. VALUE can be the
checkpoint name or number
  -s=VALUE                   stop analysis after the specified checkpoint is complete. VALUE can be the
checkpoint name or number
```

Canvas modules have the following checkpoint numbers: 1 for CanvasBin, 2 for CanvasClean, 3 for CanvasPartition, 4 for CanvasSNV and 5 for variant calling (Somatic or Germline). Omitting –s will execute Canvas until the final step in the workflow has completed. For example, specifying -c 3 in the whole-genome somatic workflow will run CanvasPartition, CanvasSNV and CanvasSomaticCaller. This could be advantageous if users want to rerun Canvas with different partitioning algorithm without the need to repeat time consuming I/O or coverage binning operations.

Ease of module reuse is further expanded by the fact that Canvas intermediate files are saved as bed files, facilitating access by third-party software such as bedtools.

Somatic Outputs

Vcf file

The primary output from CanvasSomaticCaller is a .vcf file, generally named SampleName_S1.CNV.vcf. (Later in the tumor/normal workflow, this output gets merged together with output of the SV caller, Manta, into a combined vcf file SampleName_S1.SV.vcf).

The .vcf file header includes the estimated tumor purity (1.0 for pure tumor), the estimated overall ploidy (mean copy number across all bases; 2.0 for normal data), and the estimated chromosome count (total number of chromosomes weighted by their copy number). Both OverallPloidy and EstimatedChromosomeCount are (as of 1.7.0) based on all PF calls (records with PASS in the FILTER column in the .vcf file). Example:

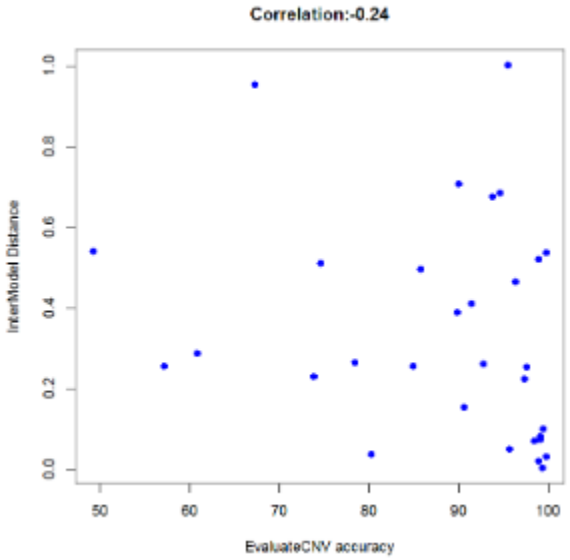
```
##EstimatedTumorPurity=0.28
##OverallPloidy=2.45
##EstimatedChromosomeCount=61.91
```

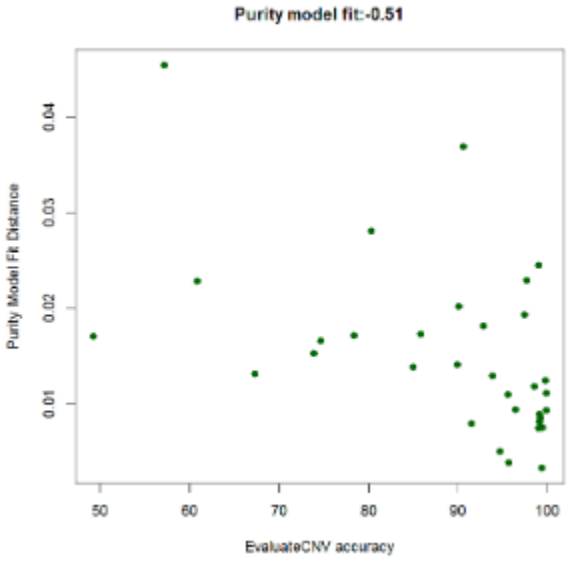
The header also includes an indication of the goodness of fit of the coverage/purity model to the data; lower values are better and should indicate more reliable calling overall:

##PurityModelFit=0.0259

##InterModelDistance=0.6853

PurityModelFit is defined as a total deviation D of the best model. InterModelDistance shows genome edit distance between the best model and other top models (defined by MaximumRelatedModels). The premise is that if the top models provide widely different genome baseline (leading to high interModelDistance), the overall modeling approach might be more unstable. Distribution of these parameters on the https://git.illumina.com/Bioinformatics/CNAT/tree/master/CanvasSomaticTestData is shown below.





PurityModelFit is more predicative of overall model quality than InterModelDistance. By selecting 70% genomewide accuracy as a threshold for high vs, low quality predictions, at inter model distance above 0.8 false discovery rate (FDR) is 50% and below 0.8 - 10%. While few observation with very high inter-model distance exist, **InterModelDistance threshold of 0.8** seem to be discriminative of poor model fit. PurityModelFit provides even better resolution - at the same 70% genomewide accuracy cut-off, and **PurityModelFit threshold of 0.022** leads to FDR of 7%. These metric value cut-offs are therefore recommended for use when assessing quality of the Canvas somatic model fit.

The copy number (CN) and major chromosome count (MCC) are indicated in the per-sample data. Below are three sample lines: The first is for a region of LOH (copy number 2, major chromosome count 2); the second is for a copy number variant (copy number 4, major chromosome count 2); the third is for a reference call (copy number 2, major chromosome count 1).

1 753815 Canvas:GAIN:1:753816:813096 N <CNV> 10 PASS **SVTYPE=LOH**;END=813096 RC:BC:**CN:MCC** 87:18:**2:2**

3 26664138 Canvas:GAIN:3:26664139:32101159 N <CNV> 61 PASS **SVTYPE=CNV**;END=32101159 RC:BC:**CN:MCC** 118:6520:**4:2**

5 1000000 Canvas:REF:3:1000000:2000000 N . 61 PASS END=2000000 RC:BC:**CN:MCC** 118:6520:**2:1**

One minor note on formatting: According to the VCF spec, if any of the ALT alleles is a symbolic allele (an angle-bracketed ID String “<ID>”) then the padding base is required and POS denotes the
coordinate of the base preceding the polymorphism. So, for non-reference calls (i.e. records where ALT is not equal to .), the POS column stores the base **before** the variant.

Merged SV/CNV vcf file

Currently, we generate separate SV and CNV calls from Manta and Canvas. The outputs (.vcf files) are then reconciled into a single merged SV vcf file, assuming both SV and CNV calling are included. This merge is done in both the Resequencing (WGS) and Tumor/Normal workflows. The logic for doing this merge is as follows:

- Keep all the .vcf header lines from both the Manta and Canvas outputs (excluding duplicates, and keeping the column headers from the Manta output)
- Add a ##source line with **IntegrateMantaCanvasVCF** to indicate that merging was done
- Canvas calls with length <= 10kb receive filter CLT10kb
- Manta DEL or DUP calls with length >= 10kb receive filter MGE10kb
- Manta calls which overlap a Canvas call receive info tag ColocalizedCanvas

The final output file name is NormalName_TumorName_G#_P#.somatic.SV.vcf, where G is the **group** index, and P is the **pair** index. Example filename: HCC1187BL_HCC1187C_G1_P1.somatic.SV.vcf

Additional outputs

CNV-CoverageAndVariantFrequency.txt - Summarized coverage and MAF data across the genome, suitable for plotting. The fields for each row of this text file are: Chromosome, start, end, copy number, major chromosome count, median hits for bins in this region, normalized coverage (where 2 = copy number 2), median minor allele frequency, reference ploidy, and a histogram of variant allele frequency (b allele frequency) for SNVs in this region. (The collection of SNVs is typically taken from the collection of heterozygous SNVs in the normal .vcf file). This table does not report coverage for regions where very few (or no) bins are available - the centromeres (and other intervals included in the filter .bed file for the reference genome) are excluded entirely from CNV calling, and so will not have any coverage or b allele frequency reported (these columns are left blank).

CNVModeling.txt - (MAF, Coverage) data points for segments and for the model. Tab-delimited file with two tables. The first gives the variant frequency and coverage coordinates (MedianMAF, MedianHitCount) for the model. The remaining table has one record for each segment used in model fitting. The values per segment are: MedianMAF, MedianHitCount, Deviation, chromosome, start, end, length, truth set copy number.

CNVConfusionMatrix.txt - available if (and only if) a truth set was specified up-front, for developer testing. Provides a confusion matrix with actual (truth set) copy number calls in the rows, and called copy numbers in the columns; the matrix entries are the number of bases with the corresponding (TruthSet, CanvasCall) copy numbers.

Working example

Here we provide a working example of how to run Canvas using aligned data from Nextera Rapid Capture Exome enrichment experiment (HCC1187 cell line). First, download the following files from https://basespace.illumina.com/analyses/29429223 and save them in the input_folder:

- `HCC1187BL_S1.bam = normal bam`
- `HCC1187BL_S1.bam.bai = normal bam index`
- `HCC1187C_S1.bam = tumor bam`
- `HCC1187C_S1.bam.bai = tumor bam index`

- `HCC1187BL_S1.vcf = SNV calls for normal bam`
- `NexteraRapidCapture_Exome_TargetedRegions_v1.2Used.txt = Nextera manifest`

Assuming that Canvas was installed into Canvas_folder and the output is written to the output_folder, execute the following command line:

```
/path/to/mono Canvas_folder/Canvas.exe Somatic-Enrichment -b input_folder/HCC1187C_S1.bam
--control-bam=input_folder/HCC1187BL_S1.bam
--manifest=input_folder/NexteraRapidCapture_Exome_TargetedRegions_v1.2Used.txt
--b-allele-vcf=input_folder/HCC1187BL_S1.vcf -r input_folder/UCSC/hg19/Annotation/Canvas/kmer.fa -g
input_folder/UCSC/hg19/Sequence/WholeGenomeFasta -o iutout_folder -f
input_folder/UCSC/hg19/Annotation/Canvas/filter13.bed --sample-name=HCC1187_NexteraEnrichment .
```

References

Roller E., et. all (2016). Canvas: versatile and scalable detection of copy number variants. *Bioinformatics,* doi: 10.1093/bioinformatics/btw163.

Olshen AB V. E. (2004). Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics*, **5** *,* 557-572.

Fryzlewicz P. (2007). Unbalanced Haar Technique for Nonparametric Function Estimation. *Journal of the American Statistical Association,* **102***,* 1 318-1327.

Rodriguez A. et. all (2014). Clustering by fast search and find of density peaks. *Science*, **6191**, 1492-1496.

Rousseeuw P. (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics*. **20.** 53-65.

Skellam J. (1946). The frequency distribution of the difference between two Poisson variates belonging to different populations, *Journal of the Royal Statistical Society*, **109**, 296.

Venkatraman ES, O. A. (2007). A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*, **31** *,* 657-663.