Canvas Design Document

Canvas 1.3.4.0

Contents

Introduction 2
CanvasBin 3
CanvasClean 5
  Single Point Outlier Removal 5
  Physical Size Outlier Removal 5
  GC Content Correction 5
  Normalization of FFPE samples 7
    FFPE de-noising algorithm 8
    Fragment-based normalization 9
CanvasPartition 10
CanvasSNV 10
CanvasDiploidCaller 11
CanvasSomaticCaller 12
  Somatic model 12
  Copy number assignment 14
Output 15
  Vcf file 15
  Additional files 15
References 16

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 genome that are present in zero, one, or more than two copies. 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. In addition, genome- and sample-wide parameters such as ploidy, purity and heterogeneity are also predicted.

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

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 data

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.

```
[~]$source UpdateEnvironmentMono
[~]$mono Canvas.exe Somatic-WGS -h
Canvas 1.3.4.0 Copyright © Illumina 2015
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
```

Canvas workflows are composed of a number of distinct steps. Using a single executable described above will launch the appropriate steps with default parameters. These include:

(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)

While using a Canvas workflow executable is recommended, users can invoke individual modules, below is an example command for CanvasBin

CanvasBin

```
mono CanvasBin.exe
>>>Command-line arguments:
CanvasBin 1.3.4.0
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
```

Below is a description of individual modules.

CanvasBin

CanvasBin creates genomic windows, or bins, across the genome and counts the number of observed alignments that fall into each bin. Canvas reads BAM files and applies a number of filters during read processing. First, only reads from proper pairs that align to the forward strand are kept. Second, CIGAR string must begin with at least thirty-five matches. To invoke this behavior at the command-line, the “-p” flag must be used when calling CanvasBin.

CanvasBin stores observed alignments in RAM as a collection of byte arrays, one for each chromosome. As the BAM file is read in, CanvasBin records the position of the left-most base of each alignment within the appropriate chromosome byte array. After all alignments in the BAM file have been read, the byte arrays have a nonzero integer where an alignment was observed and a zero everywhere else. In addition to a BAM file, CanvasBin accepts FASTA **kmer.fa** file as input (provided with Canvas distribution) that contains genomic sequences 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.

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 and non-unique positions are marked with ones and zeros respectively. The tool **FlagUniqueKmers** from Canvas distribution can be used to generate this uniqueness-flagged annotation for a reference FASTA file. 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 (-f, --filter=VALUE). This effectively excludes the region from further analysis and is therefore a mean to ignore problematic areas of the genome. A default masking file is provided in Canvas distribution (filter13.bed) that includes telomeres and centromeres (and some pericentromeric regions).

When CanvasBin is invoked, the user supplies a desired average number of alignments per bin for diploid regions (-d, --bindepth=VALUE). By default this value is set to 100 alignments per bin. This default was chosen to provide roughly the same signal-to-noise ratio as aCGH arrays. This value is then converted into a “bin size.” In a typical usage, “bin size” would refer to the size of some fixed genomic window. Here we use it to refer to the number of unique genomic 35-mers per bin. Because some regions of the human genome are more repetitive than others, physical sizes of each Canvas bin (in genomic coordinates) will not be identical. 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 is that we now expect to have the same number of reads mapping to each bin, regardless of bins “mappability” or “uniqueness”. Optionally, CanvasBin can take a Nextera manifest file (-t,--manifest=VALUE). 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.

The output of CanvasBin is a **G1_P1_0.binned** gzipped BED file containing chromosome, genomic start, genomic stop, observed counts and GC content for each bin. For illustration, the first five lines from such a file are 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 includes four procedures for removing outliers and systematic biases from the count data computed by 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. We define two bin counts, *a* and *b*, as significantly 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 test statistic:

$\mu = 0.5a + 0.5b$

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

A value of χ^2 greater than 6.635, which is the 99th percentile of the Chi-squared distribution with one degree of freedom, is taken as a threshold for bin exclusion.

Physical Size Outlier Removal

While bins in the Canvas workflow will not have the same physical (genomic) size, their average for a whole genome sequencing run will be approximately 1kb with default settings. However, some bins may be several megabases in size if they cover repetitive regions of the genome, such as centromeres and telomeres. The counts in these regions tend to be unreliable so CanvasClean removes bins with extreme physical size. Specifically, it calculates the 98th percentile of observed physical sizes and removes bins whose sizes are larger than this threshold.

GC Content Correction

One of the main factors contributing to bin count variability is the GC contents. CanvasClean provides a mechanism for correcting GC content biases that may be present in the counts calculated by CanvasBin. An example of such biases is presented in Figure 1.

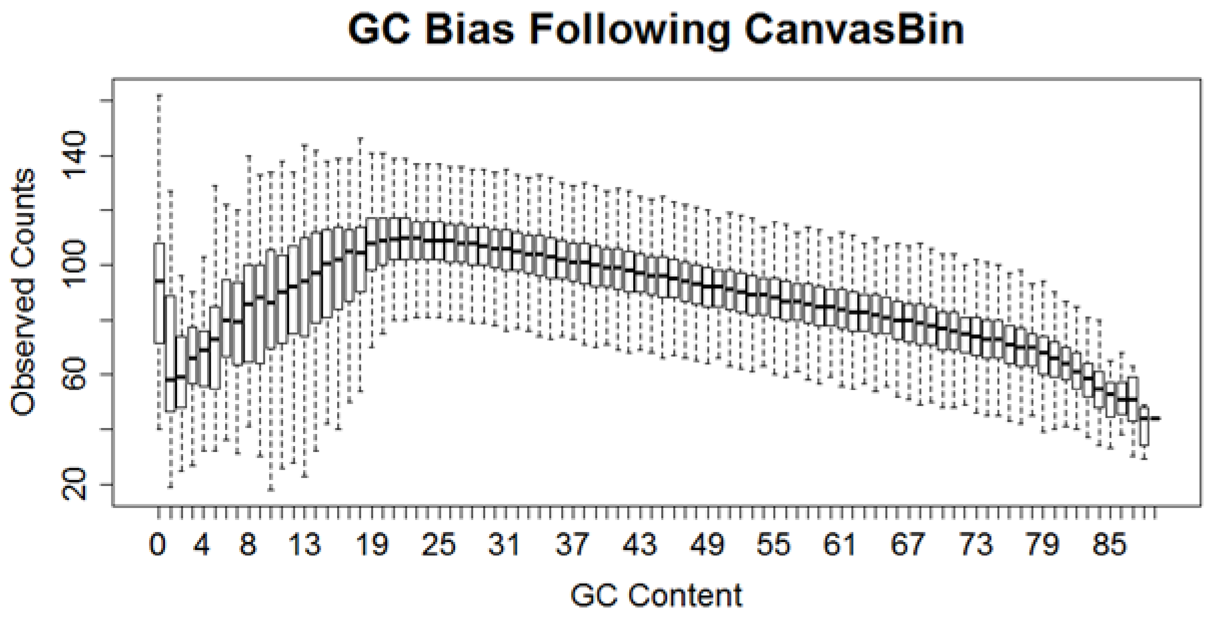


Figure 1. *Example of bin counts frequency distribution based on GC content in CanvasBin*

The correction is based on median averaging. 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, which decreases the confidence of median estimations. To prevent this, CanvasClean discards any bin where the number of bins sharing the same GC content is fewer than a threshold computed as *max(minNumberOfBinsPerGCForWeightedMedian, 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 minNumberOfBinsPerGCForWeightedMedian can be set using command line argument -w <number of bins>.

For exome workflows, CanvasClean takes a Nextera manifest file (-t,--manifest=VALUE). In this case only on-target bins are used to estimate the count distribution at each GC content level. Similarly to the whole-genome case, Canvas discards 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 computed when there are less than 100 bins for a GC value.

For some sample preparation schemes, GC content correction has a dramatic effect. Figure 2 below illustrates this for a low depth sequencing experiment using the Nextera library preparation.

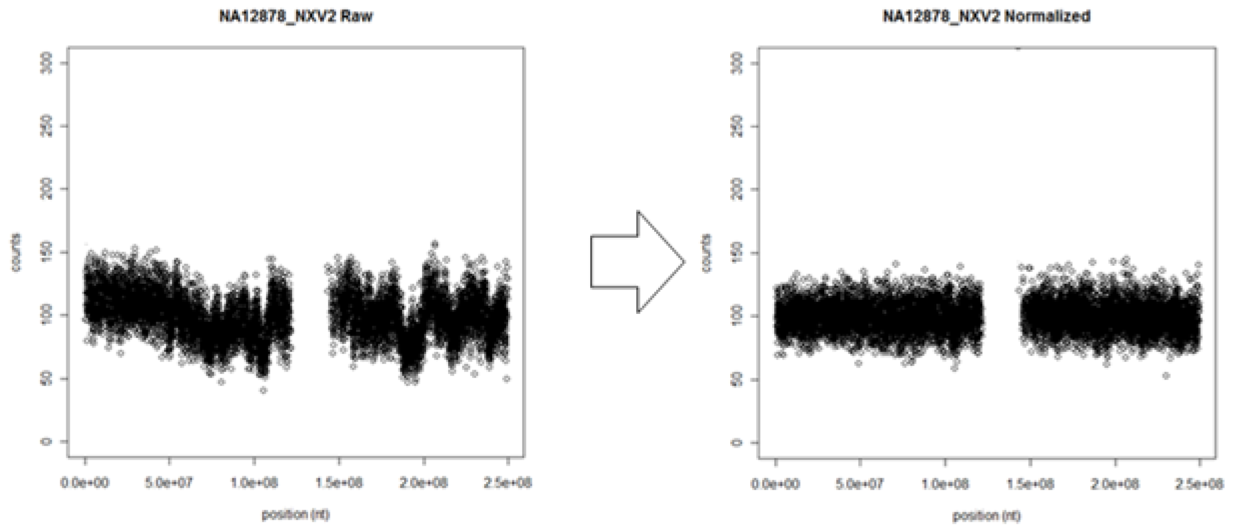


Figure 2. *Effect of GC-content normalization on coverage profile in CanvasClean. The figure on the left shows bins counts as a function of chromosome position before normalization. The figure on the right shows the same, following GC content correction.*

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 does not dampen signals from CNVs (Figure 3).

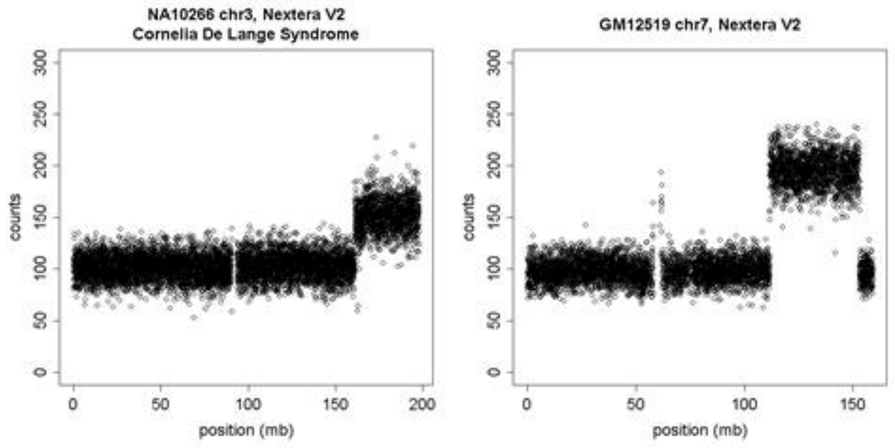


Figure 3. *Example of GC correction in samples with copy number changes.*

Normalization of FFPE samples

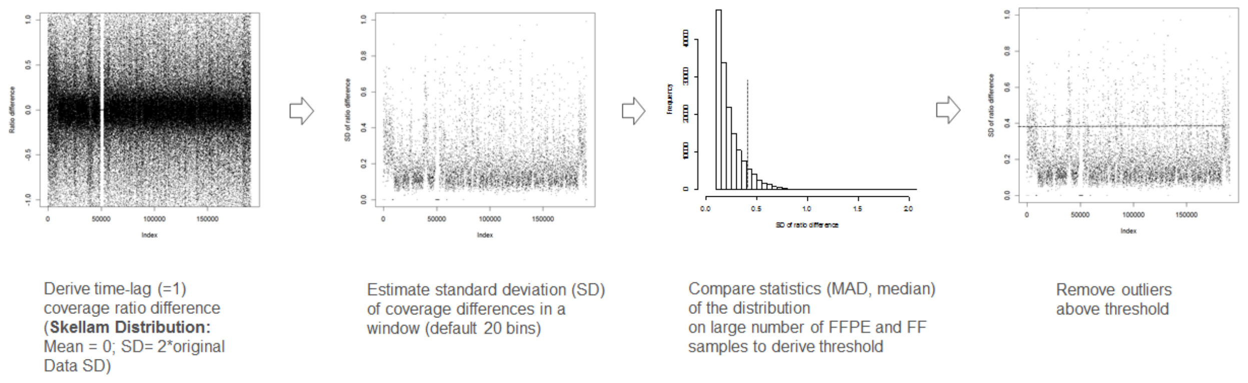
FFPE procedure is a common way of preserving tumor tissues following a cancer surgery and is therefore a frequent source of material for tumor sequencing. However, FFPE fixation introduces noise and biases during sample preparation (fragmentation, need of high temperatures for DNA extraction) that lead 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, such as chromatin confirmation 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 shown in Figure 4):



Ratio difference
Index
SD of ratio difference
Index
Frequency
SD of ratio difference
SD of ratio difference
Index
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

Figure 4. *Outline of FFPE de-noising algorithm.*

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:

- The mean for all regions (including large scale CN changes) is zero;
- The variance is doubled (Skellam J., 1946) - making regions of noise easier to detect
- 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. CanvasClean 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).

3) Comparing localSDs of coverage data from a reference set of FFPE and FF samples derived from the same individuals shows a clear separation of FFPE and FF groups by the metric (localSD threshold is

set to 10 from examples in Figure 5). This property is used to set a localSD threshold specifically to insure that only noisy FFPE samples undergo de-nosing.

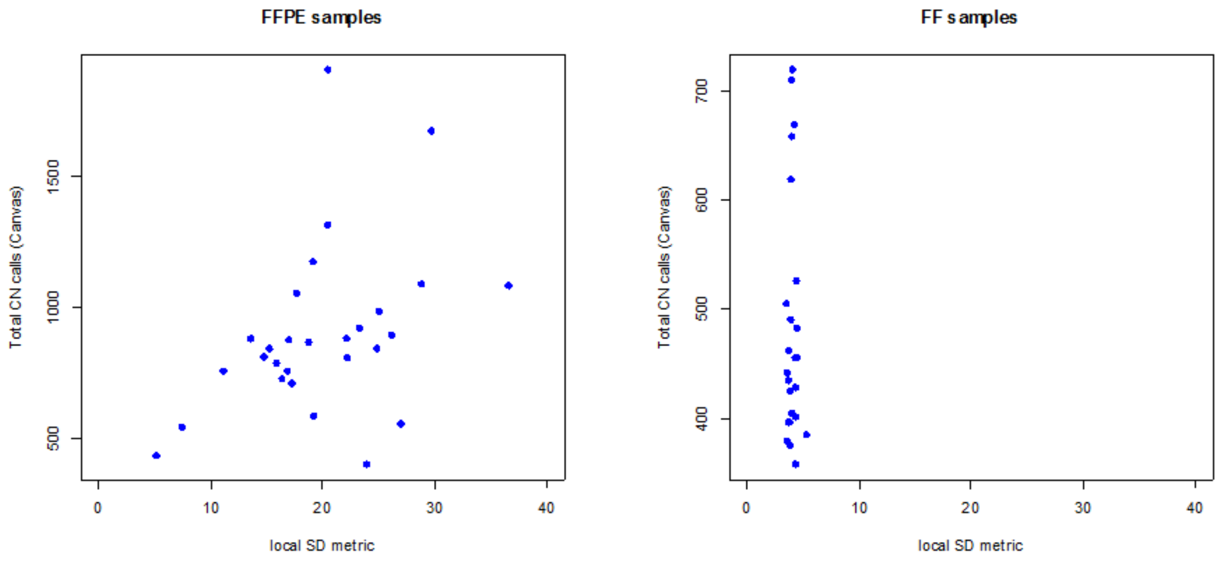


Figure 5. *Distribution of localSD metric in FFPE and FF samples.*

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 latter is particularly influenced by a PCR amplification step occurring during FFPE library preparation. 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 (Figure 6). Fragment-based normalization can be disabled by specifying --custom-parameters=CanvasBin,–mode=TruncatedDynamicRange argument on Canvas executable command line.

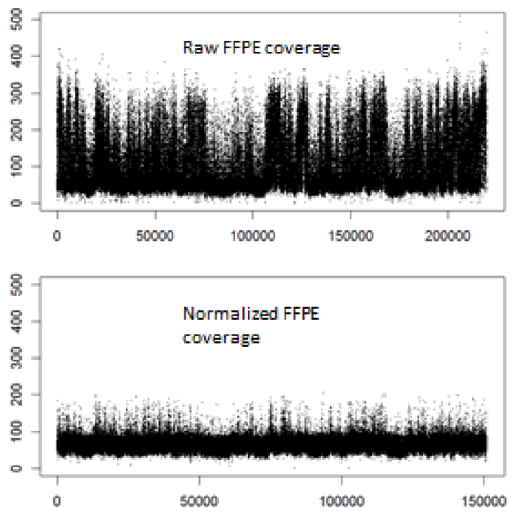


Figure 6. *Example of fragment-based GC correction on a sub-chromosome coverage interval.*

CanvasClean output has the same format as that written by CanvasBin.

CanvasPartition

CanvasPartition takes normalized CanvasClean BED file as input and segments all bins into a set of distinct segments having equal mean coverage values. CanvasPartition implements a number of segmentation algorithms including unbalanced Haar wavelets (the default) and circular binary segmentation (CBS).

Haar wavelet segmentation is a partitioning method introduced by (Fryzlewicz P, 2007). It is based on a traditional discrete wavelet transform (DWT), but in contrast to DWT allows for an arbitrary segment size. This is achieved by using adaptation of matching pursuit algorithm for selection of breakpoint basis vectors in O(Nlog(N)) time. The resulting DWT coefficients along with original Haar basis vectors can be used to reconstruct de-noised signals after setting all DWT coefficients below a user specified threshold to zero. Threshold recommended in (Fryzlewicz P, 2007) is used by CanvasPartition. CBS 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 between bins are maximally different. 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 breakpoints. The computational complexity of the algorithm $O(N^2)$ so the problem is divided into sub-chromosome problems followed by merging. Heuristics are also used to speed up the permutation testing (Venkatraman ES, 2007).

CanvasPartition outputs a gzipped BED 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 such a 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

The CanvasSNV step is used to estimate SNV allele frequencies (aka minor allele frequencies, MAF). It takes as input (1) VCF file with variant calls or the normal reference sample or path to a dbSNP reference VCF file, and (2) the aligned reads (in BAM format) for the tumor/reference sample. For the tumor/normal workflow, germline SNVs from the normal sample are used to identify all heterozygous SNVs in tumor (GT = 0/1 or 1/0) that pass filters (PASS in the FILTER column). We also require that the GQX tag (if present) is larger or equal 30. For germline calling, CanvasSNV can use either the germline SNV calls or has the option to review all dbSNP sites. For each SNV site, Canvas computes the number of alignments made to the REF and ALT alleles. This calculation considers all reads which are not flagged as PCR duplicates, secondary or supplementary alignments. Next, for all reads whose alignment to the reference covers a site of interest, CanvasSNV estimates how many reads have the reference base, and how many reads have the alternative base. Given A and B read counts 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 lower and will approach 0.5 as coverage increases). 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 #) and record lines that 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, σ, 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 σ/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 not considered.

Following assignment of copy number states, neighboring regions with the same copy number call are merged into a single region.

Finally a Phred-like *q*-scores are assigned to each region. To derive them, a logistic regression 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. loss, gain or no change. The three features used by the model are:

- *LogBinCount* $\log_{10}$(1+number of bins). Segments with larger number of bins can be assigned a copy numbers more confidently.
- *ModelDistance*: Distance between observed Coverage, and MAF values and the ones predicted by the selected model. Lower distances reflect a better fit.
- *DistanceRatio:* Ratio between model distance of 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 correct.

The probability is then scaled to generate a Phred-scaled likelihood. The coordinates of non-diploid regions and their *q*-scores are written to a VCF file. Two filters are applied to when assigning a PASS flag to a variant. First, a variant must have a *q*-score of 10 or greater. Second, a variant must be of size 10kb or greater.

CanvasSomaticCaller

CanvasSomaticCaller is used in tumor-normal workflow to identify regions with an abnormal copy number (e.g. for autosomes the copy number is not 2 or b-allele frequency is not ~50%) and to estimate global tumor characteristics, such as genome-wide ploidy and sample purity. The latter are crucial to the overall quality of individual CNV calls: any mistakes in ploidy and/or purity predictions tend to have a detrimental genome-wide effect on most variant calls. Before describing details of the CanvasSomaticCaller workflow, key principles of the somatic model are introduced first.

Somatic model

We begin by briefly listing key design principles and definitions of Canvas somatic model.

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.** For regions with a given **ploidy** (copy number and major allele count), minor allele frequency for each SNV can be theoretically derived

- `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.)`

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 and purity values, machine learning techniques can be used to infer best set of features that discriminate between correct and erroneous somatic models.

CanvasSomaticCaller also attempts to estimate overall variant heterogeneity if the tumor is polyclonal. While finding ploidy and 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.

Given these definitions, the major steps of CanvasSomaticCaller workflow are described below in Panel 1 and Figure 6.

Panel 1. Canvas somatic model workflow
1. Load coverage segments identified by CanvasPartition.

2. Load the 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. The following criteria for MAF selection are used:
    1. CanvasSomaticCaller filters 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.
    2. If the number of MAFs before filtering is greater than the given MAF count threshold (default: 50) but less than this threshold after filtering, the threshold is lowered to be the number of MAFs after filtering.
3. For each segment identified by CanvasPartition, derive the median number of counts per bin ("coverage") and the median minor allele frequency (MAF).
    1. Each segment is assigned a weight, which is used in computing the deviation described below. The weight is equal to 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 there are fewer than ten MAFs in a segment, the weight is multiplied by the ratio of the number of MAFs over 10. This is due to the fact that a smaller number of MAFs will provide a less reliable estimation of the true median MAF for the segment. Such segment weighting technique makes model fitting more robust to noise.
4. Iterate over diffident values of diploid coverage D and the overall purity p to derive model deviation.
    1. Given a candidate tuple (D, p), compute the expected (MAF, Coverage) points for each copy number value.
    2. The **model deviation** for (D, p) is the sum, across all segments, of **||**(SegmentMAF, SegmentCoverage) - (ModelMAF, ModelCoverage)**||**.
5. Estimate a **model penalty term**. To derive it, percent normal, percent CN=2 and diploid distance variables were calculated from preliminary genome-wide copy number calls for each model. They are then used as predictor variables in logistic regression model trained on a set of 58 tumor/normal pairs with manually assigned ploidy and purity values. Regression coefficients from this model were transformed into new set of weights for the final penalty term as follows:
    1.
        1. *0.25*DiploidDistance*: Scaled number of CN events needed to transform a given genome into diploid
        2. *+0.35*CN2: Percentage* of CN = 2
        3. *+0.3*PercentNormal*: Percentage of normal CNs
6. Somatic caller also attempts to account for tumor polyclonality. For this, each segment tuple (SegmentMAF, SegmentCoverage) is assigned to distinct clusters using the EM algorithm. The clusters are initialized by sampling (SegmentMAF, SegmentCoverage) values from observed segments that have been ranked by k-nearest neighbor distance (where k is set to the total number of segments). Only top 70% segments are used. This value is that to favor initial EM values that fall near clusters associated with major copy number changes. Optimal number of clusters was selected using Silhouette coefficient (Rousseeuw P., 1987). **Cluster deviation** for cluster (D, p) is then defined as the sum of distances between segments belonging to the cluster and the closest expected (MAF, Coverage) tuple for a given model as calculated in step 4. In general, clusters located farthest from expected (MAF, Coverage) centroid are more likely to be heterogeneous: if cluster deviation > total deviation * 1.5, the cluster is not considered when calculating overall cluster deviation (value optimized on training corpus).
7. The **total deviation** is than defined as the average of model deviation, cluster deviation and penalty term. Coverage (ploidy) and purity model with the lowest total deviation is used to assign copy number calls.

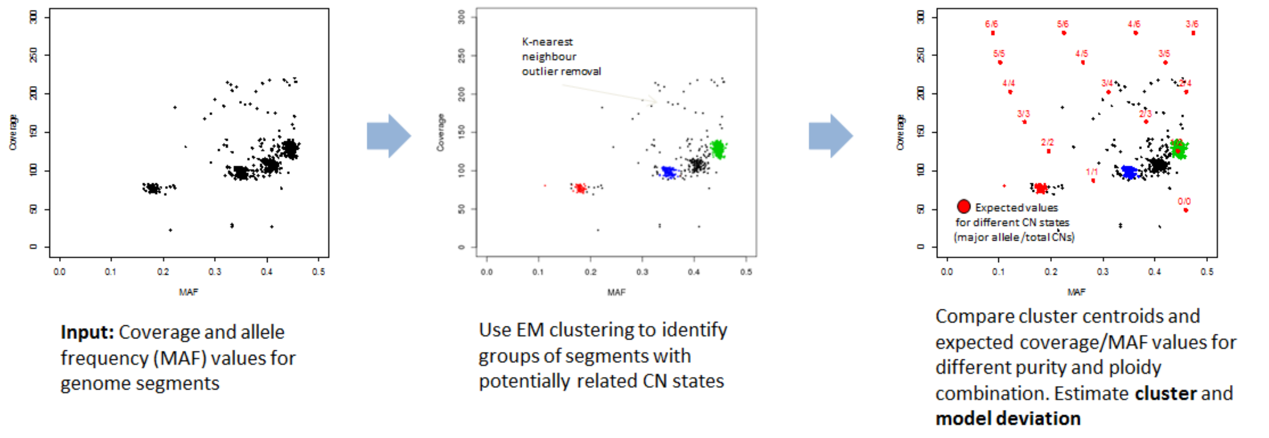


Figure 6. *An example of Canvas somatic model workflow. Coverage and MAF values are clustered in a two dimensional space using EM algorithm after outlier removal. Expected values of coverage and MAF for each copy number state (represented as major allele/total CNs on the plot) are also calculated for a current ploidy model. The difference between expected and observed values and cluster centroids is then used to calculate total and cluster deviation as described in the Canvas somatic model workflow panel.*

Copy number assignment

Once preliminary calls are made, a **merge** step is carried out. Adjacent segments with the same copy number values 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 filter BED file) between them. Next, CanvasSomaticCaller assigns Phred-like *q*-scores to each variant. A logistic regression 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. losses, gains or no change. The three features used are:

1. *LogBinCount* $\log_{10}$(1+number of bins). Segments with more bins can be assigned a copy number more confidently
2. *ModelDistance*: Distance to the model. (Lower distances reflect a better fit)
3. *DistanceRatio:* Ratio 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-like 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 main output of CanvasSomaticCaller is a variant call file (VCF 4.1 format) listing all intervals which were copy number was assigned. Copy numbers are reported as a REF, LOSS, or GAIN depending on the copy number count and expected count. The expected count is generally two (copy number 2 is REF, copy number 0 or 1 is LOSS, copy number 3 or greater is GAIN). The exception is when a ploidy BED file is passed to Canvas, it can update the expected number of copies for particular chromosomes. For instance, for chromosome X and chromosome Y from an XY sample, copy number calls of 2 on X or Y would be called as a GAIN, with the exception of PAR regions that still have an expected copy number of two.

Output

Vcf file

The primary output from Canvas is a VCF file named **CNV.vcf.** VCF 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). For 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 overall calls:

```
##PurityModelFit=0.0259
```

The copy number (CN) and major chromosome count (MCC) are indicated in the per-sample data. Below are three example lines: The first is for a region with 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 753816 Canvas:GAIN:1:753816:813096 N <CNV> 10 PASS SVTYPE=LOH;END=813096 RC:BC:CN:MCC 87:18:2:2
3 26664139 Canvas:GAIN:3:26664139:32101159 N <CNV> 61 PASS SVTYPE=CNV;END=32101159 RC:BC:CN:MCC 1
18:6520:4:2
5 1000000 Canvas:REF:3:1000000:2000000 N . 61 PASS END=2000000 RC:BC:CN:MCC 118:6520:2:1
```

Additional files

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), 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).

TempCNV folder - Contains intermediate files from individual Canvas workflows described above and some additional temporary files from CanvasSomaticCaller described below.

CNVModeling.txt - Tab-delimited file with two tables representing (MAF, Coverage) data points for segments and for the model. The first gives the variant frequency and coverage coordinates (MedianMAF, MedianHitCount) for the selected best 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 and length.

References

Olshen, AB, *et al*. (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***,* 1318-1327.

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