Canvas Software Design Document

Canvas 1.3.x

(marketing name: Isaac Copy Number Variant Caller)

Copy Number Calling from Read Depth and SNV Frequency Information

Software Design Description

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 a normal germline DNA sample will be diploid, that is, having two copies. Canvas seeks to identify regions of the sample's genome that are present zero, one, or more than two times in the genome. Briefly, this is achieved by scanning the genome for regions of the genome having 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 apply to low-depth cytogenetics experiments, low-depth single-cell experiments and to whole genome sequencing experiments. Since then it has been extended to handle tumor/normal samples. It has also been extended to handle some targeted data such whole exome experiments, whole exome tumor/normal data. Large targeted enrichment panels are also appropriate; however, using Canvas on targeted amplicon data (TruSeq Amplicon) is not appropriate.

Canvas for germline data can be conceptually divided into four processes:

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

Canvas for tumor/normal data has a similar set of steps:

(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, for heterozygous SNVs identified in the normal sample (implemented as CanvasSNV.exe)

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

Each of these processes will be detailed in subsequent sections.

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 Bowtie and Isaac aligners. 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 Isis re-sequencing 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 keeps in memory a collection of byte arrays to store observed alignments, one array for each chromosome. Each array's length is the same as its corresponding chromosome's length. 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 of 3 counts up to 10 hits at each position.

After reading in the BAM file, a special FASTA file is read in, one chromosome at a time. This FASTA file, provided in the Annotation/Canvas subfolder for a reference genome (and named **kmer.fa**), contains the genomic sequences that were used for alignment. 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 ensures 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 byte 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 IsisTools 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.

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 filter12.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 either a rare CNV present in the reference genome or an otherwise problematic region.

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. (This default was chosen to provide roughly the same signal-to-noise ratio as an Agilent CGH array.) 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.

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. Each bin’s chromosome, genomic start, genomic stop, observed counts and GC content are output to disk as a tab-delimited text file having one column for each of these values. 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 thee simple 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

CanvasClean takes as input the BED-like 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:

$\mu = 0.5a + 0.5b$

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

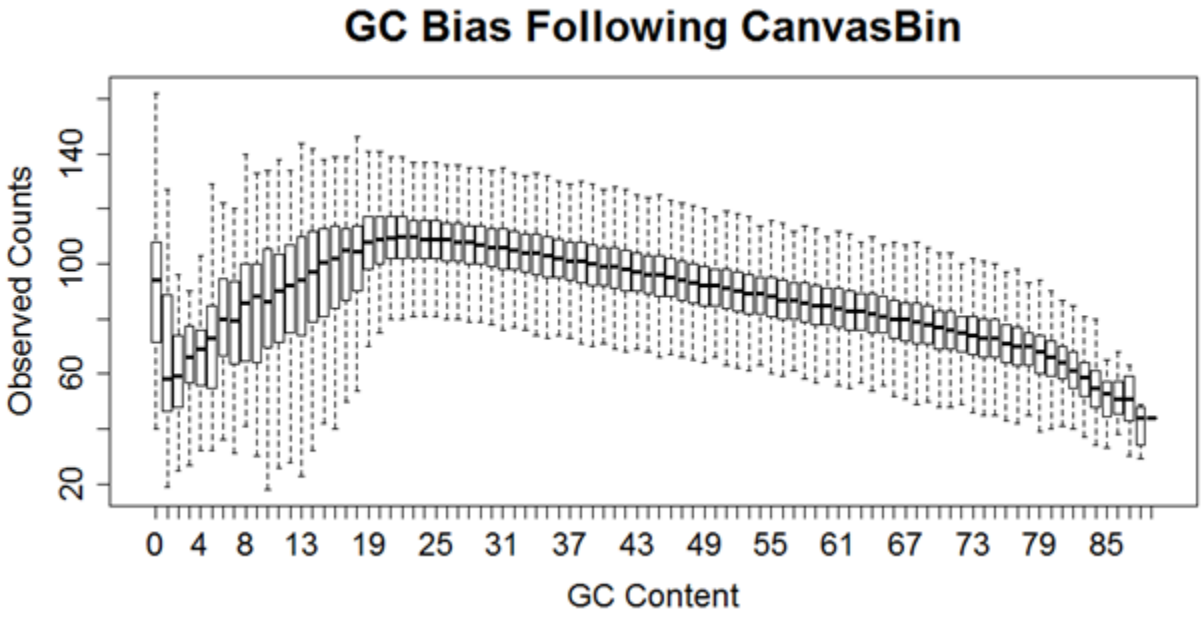
A value of ^{2} greater than 6.635, which is the 99th percentile of the Chi-squared distribution with one degree of freedom, is considered “very different.” If a bin’s count is very different from both its immediate upstream and downstream neighbors’ counts, then the bin is deemed an outlier and removed.

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.

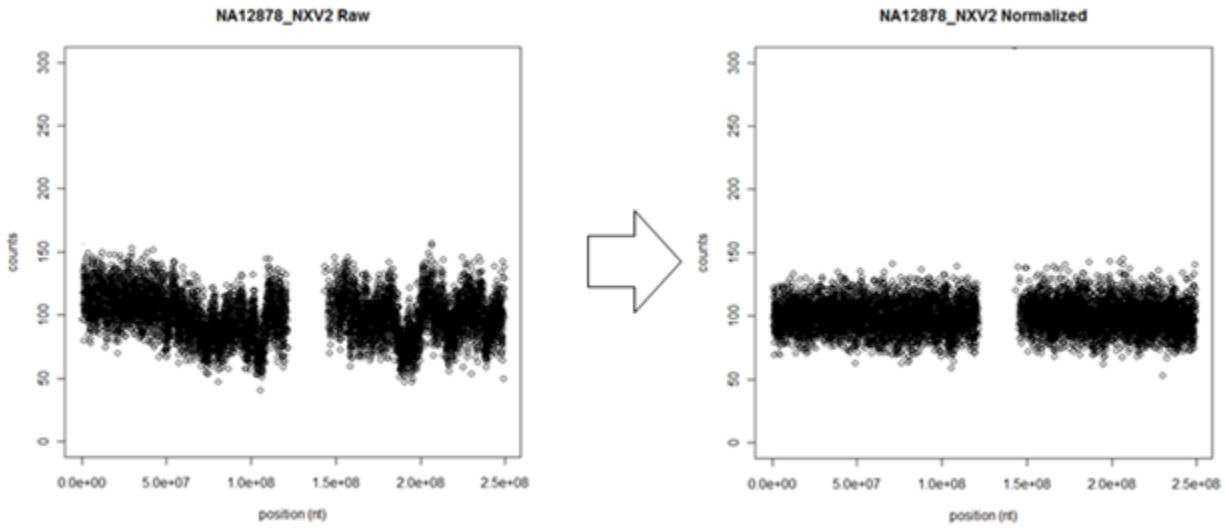


The correction is simple. Bins are first aggregated by their GC content, which has been rounded to the nearest integer. Second, each bin's count is divided by the median count of bins having the same GC content. Finally, this value is multiplied by the global median. The effect is to flatten the midpoints 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>.

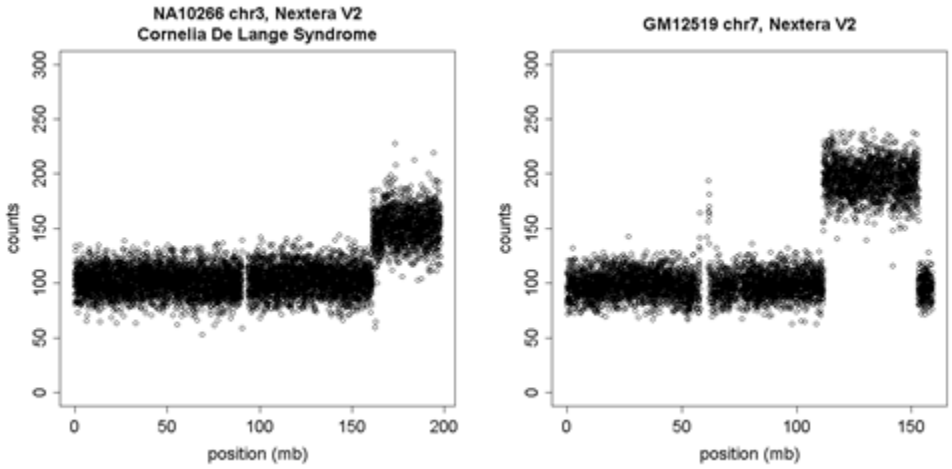
Optionally, CanvasClean takes a Nextera manifest file through command line argument -t <path to manifest> (–manifest=<path to manifest>). In this case, only on-target bins are used to estimate the count distribution at each GC content level. Similarly, we discard any any bin where the number of on-target 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.

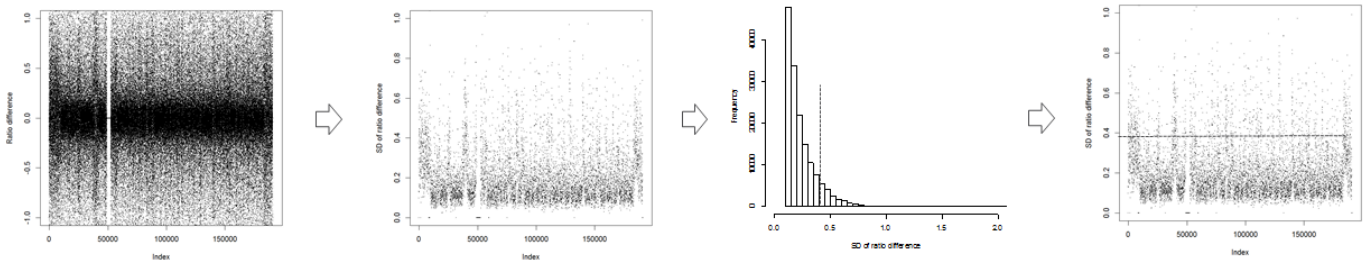
Importantly, the normalization signal does not dampen signal from CNVs. Below are two figures, one for a chromosome known to harbor a single copy gain and one for a double copy gain.



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 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

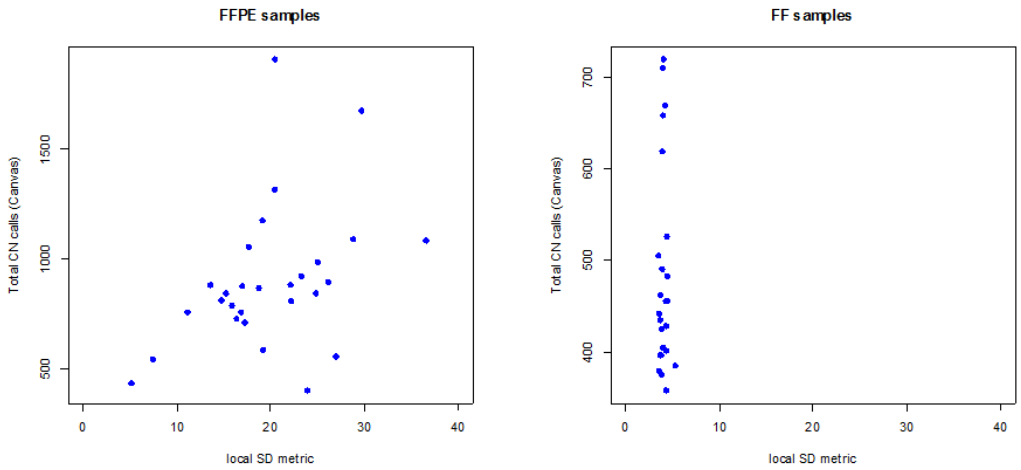
Remove outliers above threshold

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 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.



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

CanvasPartition

CanvasPartition provides implementation of a number segmentation algorithms for partitioning CanvasClean output: currently these include circular binary segmentation (CBS) and unbalanced Haar wavelet segmentation.

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 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.

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.

Another BED-like file is written to disk by CanvasPartition (e.g. **G1_P1.partitioned**). It has columns containing each bin's chromosome, genomic start, genomic stop and an integer label that identifies which partitioned region the bin belongs to. This intermediate file can be included in output using the RetainIntermediateCNVFiles sample sheet setting.

CanvasSNV

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

CanvasSNV first identifies a collection of SNVs to use: All heterozygous SNVs from the .vcf file (GT = 0/1 or 1/0), which pass filters (PASS in the FILTER column), with GQX >= 30. For each SNV site, it then computes how many observations were made for the REF and ALT alleles. This calculation considers all reads which are not flagged as a PCR duplicate, not a secondary alignment, and not a supplementary alignment. We count (for all reads whose alignment to the reference covers a site of interest) how many reads have the reference base, and how many reads have the alternative base. major copy number

Given X and Y reads for the two alleles being considered (where X+Y>0), we define the minor allele frequency (MAF) as min(X, Y) / (X + Y). 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 with name of the form **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. This file can be included in the analysis folder using the **RetainIntermediate CNVFiles** sample sheet setting. Here are sample first lines of the 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 module of the Canvas algorithm for germline data is to assign discrete copy numbers to each region identified by CanvasPartition and determine it's major copy number (MCC, copy number of major allele) where

possible. The input into the algorithm are coverage and MAF values for each CanvasPartition segments. The model itself is a Gaussian Mixture model with a separate state for different copy number and major copy number states: e.g. for CN=2 model has two states representing diploid case (MCC=1, CN=2) and LOH (MCC=2, CN=2). The means and covariance matrix 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. The mean and standard deviation are estimated by the autosomal median and MAD of counts. Following EM fitting, for each CanvasPartition segment, [CN,MCC] model tuple with the highest posterior probability is used to assign segment copy number. 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.

Phred-like *q*-scores are assigned to each region using a simple logistic function derived using array CGH data as the gold standard. The probability of a miscall is modeled as:

$$p = 1 - \left(\frac{1}{(1 + e^{0.5532 - 0.147N}}\right)$$

Where *N* is the number of bins found within the non-diploid region. This probability is converted to a *q*-score by q = -10 log p

This estimate is likely 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 applied in the tumor/normal copy number calling case, to identify regions whose different copy number and ploidy are not normal (copy number <> 2 and/or heterozygosity not corresponding to the two distinct chromosomes from the normal). The caller proceeds as follows:

- Load the coverage-based segments identified by CanvasPartition
- 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.
    - We prune the zero MAFs for a segment If more 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.
    - In case pruning is done, we further take only MAFs > 0.1 if there are MAFs > 0.1 in the segment. This is because some homozygous positions have non-zero MAFs that are very close to zero.
    - If the number of MAFs before pruning is greater than the MAF cutoff (default: 50) but is less then the cutoff after pruning, we lower the cutoff to the number of MAFs after pruning.
- For each segment identified by CanvasPartition, note the median number of counts per bin ("coverage") and the median minor allele frequency (MAF).
    - The number of MAFs of each segment must meet the MAF cutoff (default: 50) to be used in model fitting. However, the MAF cutoff is relaxed so we have at least 20 segments for model fitting.
    - 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 the less number of MAFs, the less reliable the median MAF of the segment. This makes model fitting less sensitive to noise.

- Model the median diploid coverage D and the overall purity p: Given a candidate tuple (D, p), compute the expected (MAF, Coverage) points for each ploidy. Loop over a collection of all segments of reasonable size (>50kb) and informative numbers of SNVs (>=50). For each of these segments, classify it as whichever ploidy is closest (euclidean distance) to this segment's (MAF, Coverage). The **deviation** for (D, p) is the sum, across all segments, of ||(SegmentMAF, SegmentCoverage) - (PloidyMAF, SegmentCoverage)||. The final model distance is derided by combining deviation and additional coefficient-weighted factors as follows:
    - *+0.25*DeviationScore*: Model deviation
    - *+0.25*DiploidDistance*: Scaled number of CN events needed to transform a given genome into diploid
    - *+0.35*CN2: Percentage* of CN = 2
    - *+0.3*PercentNormal*: Percentage of normal CNs

  To derive the formula coefficients, model deviation, percent normal, percent CN=2 and diploid distance were used as predictor variables in multivariate logistic regression that was trained on a set of 58 tumor/normal pairs with manually assigned ploidy and purity values. Regression coefficients from this model were then transformed into new set of weights for the final score. The model with the lowest score is selected as a preliminary model of overall coverage and purity.

- 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.
- This preliminary model is used as input to a gaussian mixture model (GMM), which models the size and covariance of each cluster to refine CNV calls for segments.
- 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)
- 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%.
- 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:
    - *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. Note that this q-scoring is different from the one described for CanvasDiploidCaller. 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)

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.

CanvasSomaticCaller 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). 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

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 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** 118:6520:**4:2**

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

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), 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.

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

Regression testing of Canvas

Some input data for CanvasSomaticCaller has been saved in TFS at $Illumina.Isis\Dev\Trunk\Src\Canvas . In particular, the subfolders CanvasSomaticTestData , CanvasTestData, CanvasGermlineTestData. This includes "truth set" data for several germline and tumor samples (.vcf files giving the actual copy numbers for various regions - these aren't perfect but they're generally pretty good), and the inputs to CanvasPartition and CanvasSomaticCaller / CanvasDiploidCaller (i.e. outputs from CanvasClean, CanvasPartition and CanvasSNV). A Python script named **RunCanvasRegressionTests** is available to run Canvas for each of several data-sets, and summarize the results. By default, it exercises the somatic test cases; pass -G on the command-line to run the germline tests instead. Here is the current usage info:

```
Options:
-G: Run germline tests (instead of somatic tests)
-t Test corpus (default: CanvasSomaticTestData\CanvasSomaticRegressionTests.txt)
```

```
-e ExecutableFolder (default: x64\release)
-g GenomeRoot (default: d:\genomes)
-p: Run CanvasPartition, too
-o OutputPath (default: CanvasSomaticTestData\Results.txt)
-x: Extra parameters to pass along to CanvasSomaticCaller
-z: Extra parameters to pass along to CanvasPartition
-v path to EvaluateCNV (default: ..\..\Tools\EvaluateCNV\bin\x64\Release\EvaluateCNV.exe)
```

Results are written to: CanvasSomaticTestData\Results.txt or CanvasGermlineTestData\Results.txt

For each folder, we note the accuracy (% of bases whose copy number call matches the truth set), and direction accuracy (% of bases where the direction of copy number change matches the truth set - here we consider a call valid if both new call and truth call are 2, both are <2, or both are >2). Direction accuracy is more forgiving (it will be >= the exact accuracy) and reflects the notion that calling an amplification is more important than pinpointing whether there are 5 or 6 copies. For more details on CNV evaluation see EvaluateCNV

References

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

Fryzlewicz P. (2007). Unbalanced Haar Technique for Nonparametric Function Estimation. *Journal of the American Statistical Association,* 1318-1 327.

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

–