Census-seq Core Computational Protocol

version 0.2.0 (7/09/20)

James Nemesh
Steve McCarroll's lab, Harvard Medical School

Introduction

The following is a manual for using the software we have written for processing Census-seq sequence data from aligned reads to donor proportional representation. This guide will not cover read alignment in depth as standard alignment pipelines produce data that can be used in this framework without any special pre-processing. This guide will cover both interpretation of Census-seq output, as well as the Roll Call and CSI programs that help troubleshoot results in cases where there may be contamination of an experiment with cells from an unexpected donor.

We may release updates to this manual as we learn from users' experiences. If a revision simply contains additional hints or advice or detail, then we will update the date on the protocol but not the version number. Whenever we implement a substantive change to the software or protocol, we will increment the version number.

We hope this is helpful and that you are soon generating exciting data with Census-seq.

Prerequisites for Census-seq [Aligned BAM + VCF]

To run Census-seq, there are two critical pieces of data needed: a VCF file containing the set of variants in the population you wish to ascertain, and an aligned BAM file that is aligned to the same reference as the VCF file containing observations of the sites in the VCF.

The aligned BAM can be generated by any number of methods. We use Picard tools to demultiplex libraries, then align fastq files with BWA. You could also use another existing pipeline to align BAM files - there are no Census-seq specific pre-processing required. When we first prototyped this software, we used Illumina's BaseSpace to perform alignment. If you have a standard protocol for aligning DNA sequence data that generates valid aligned BAMs you should be able to use that output.

The VCF file required also follows a standard format. Generating a VCF file from raw sequence reads is a more complicated process than sequence alignment, and is outside the scope of this document. What we'll focus on is data quality and preprocessing of the VCF, as this is critical to obtaining accurate results from the CensusSeq algorithms.

Preprocessing a VCF

From the paper:
Prior to running Census-seq, VCF files were processed to filter variants and add additional site-level information. Variants were first normalized to their appropriate reference sequence using BCFTools; this splits multiallelic SNPs into multiple biallelic SNPs, and sets the reference allele to be the reference base of the genome at that position. Variants that were monomorphic were dropped, as well as those without a PASS filter, where the site was flagged as problematic during VCF generation. Sites without rsID annotations were updated using information from dbSNP when possible, and otherwise site names were changed to chromosome:position:ref_allele:alt_allele. Allele frequencies calculated from the 1000 Genomes Project were annotated at all available sites.

Let's break that down into a series of commands. BCFTools is an excellent bit of software that can cover all of our needs. Below are examples of cleanup commands, using a GRCh38 reference genome. Most of these steps are optional because the same functionality can be replicated by CensusSeq software, but you may wind up parsing your VCF many times, and some of us are impatient and like to minimize wait time for results. If you only apply one step to your data, we suggest variant normalization.

Variant normalization:
This splits multiallelic variants into biallelic variants. If not used, multiallelic variants will not be considered by CensusSeq tools. This step is **optional but recommended** to include as many variants as possible in analysis.

```
bcftools norm -m -any -f /path/to/GRCh38.fasta.gz -O z -o mydata.ref_norm.vcf.gz mydata.vcf.gz
```

When run successfully, there will be a summary of the actions taken:
```
Lines   total/split/realigned/skipped:    306231/18903/22504/0
```

Remove non-passing and monomorphic variants:
Remove all variants that have fewer than one count of the alternate allele, as well as sites that don't PASS VCF QC during generation. Depending on how your VCF was generated, this can provide a small to moderate speedup for CensusSeq processing. **Optional** (CensusSeq will parse and skip all monomorphic sites for you, as well as all non-passing sites.)

```
bcftools view -f .,PASS -a -c 1:nonmajor -O z -o mydata.ref_norm.variant.vcf.gz mydata.ref_norm.vcf.gz
&& bcftools index -t mydata.ref_norm.variant.vcf.gz
```

[This also indexes your output VCF, which we'll do with all our downstream commands.]

Add rsIDs to VCF:
For variant sites at which an rsID is available in dbSNP, change the variant name to the rsiD name. For sites at which no variant site is available, change the variant name to be contig:position:ref allele: alt allele. **Optional**, but nice for downstream analysis. Find the dbSNP VCF here.

```
bcftools annotate -a dbsnp.vcf.gz -c +ID -O v mydata.ref_norm.variant.vcf.gz | bcftools annotate --set-id
+'%CHROM:%POS:%REF:%FIRST_ALT' -O z -o mydata.ref_norm.variant.rsid.vcf.gz -
```

Add reference population allele frequencies
For each variant, add the 1000 genomes allele frequency information. The motivation for why you might want to run this step will be discussed more in depth in the CSI Analysis section. You can find 1000 genomes site information here. We suggest concatenating the contig level VCFs together via bcftools concat, then annotating the sites in the same manner as you do your own reference VCF.

```
bcftools annotate -a ALL.GRCh38_sites.20170504.ref_norm.vcf.gz
-c CHROM,POS,REF,ALT,INFO/EUR_AF  -Oz -o mydata.ref_norm.variant.rsid.1kg_AF.vcf.gz
mydata.ref_norm.variant.rsid.vcf.gz && bcftools index -t mydata.ref_norm.variant.rsid.1kg_AF.vcf.gz
```

VCF quality control

Below is a list of filters that we use at the sample, variant, and genotype level to QC VCF files before we run CensusSeq. We will be releasing a document dedicated to VCF cleanup best practices that we employ in the near future along with an ipython notebook containing template code for cleaning up your VCF.

Filter Genotypes:

- Allele balance filter (AB = AD/sum(AD))
  - Homozygous ref site: >10% alt reads
  - Homozygous alt site: >10% ref reads
  - het site: <25% ref reads or >75% ref reads

Filter Variants:

- Variant Quality Score Recalibration != PASS
- Allelic Depth (<= 10)
- Genotype Quality (<= 20)
- HWE error rate < 1e-3
- Call rate <= 50%
- remove variants in low complexity regions
- remove variants in segmental duplications. (LCR and SEGDUP can be found in UCSC database)

Samples with ambiguous sex can be removed prior to VCF generation (Male: chromosome X F-stat < 0.8 and Female: chromosome X F-stat > 0.4)

Filter Samples:

- Remove twins or sample duplicates
- Call rate: (look at distribution; typically remove samples with call rate < 0.95)
- Depth of sequencing (remove samples with DP < 10)

Variant Annotation:
Add gnomAD Allele Frequencies
Update missing rsids using gnomAD

Filter monomorphic sites

Census-seq

Overview

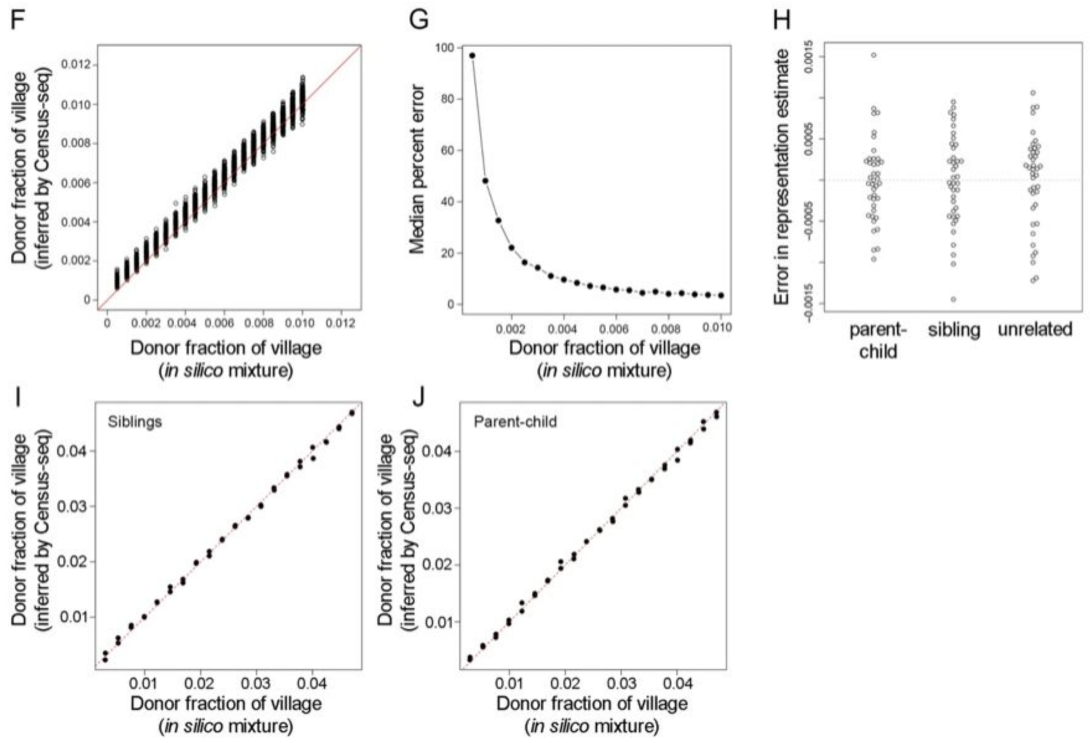
The goal of Census-seq is to measure the contribution of each donor to a cell pool – both to monitor population dynamics, and for quantitative phenotyping. We do this systematically, routinely and inexpensively, without the need for single-cell analysis, simply by lightly sequencing genomic DNA from the cells. The donor mixture determines what ratio of alleles are present at every SNP. We developed a gradient-descent algorithm to find the donor-mixing coefficients that maximize the likelihood of any observed sequence data. We find that our approach can accurately characterize pools with dozens of donors, enabled by this “Census-seq” approach, we now profile pools at every passage and measure donor representation as a quantitative phenotype.

Since Census-seq is an optimization problem that finds the best mixture of a given set of donors to explain the data, it requires a specific list of donors that are believed to be in the pool. Census-seq is accurate when the proper set of donors is tested, but becomes less accurate when unexpected donors are present in the pool. When the donor list is inaccurate, reads that don’t belong to any of the true donors in the pool tend to be assigned uniformly to all of the donors in the pool, leading to a more uniform distribution of donor proportions. If the donor list diverges significantly enough from the true donors in the pool the algorithm may not converge.

To deal with this problem, we’ve also included a pair of programs that can be used to provide quality control on the input pool and donor list. Roll Call can detect the issues of sample swap effects (where another known sample in the population was introduced to the pool), while CSI detects contamination (where a donor outside the known set of donors was introduced to the pool.) These programs will be discussed in detail later in the document.

Census has been tested against both unrelated individuals as well as trios. Provided the correct list of donors is supplied, census is accurate down to ~ 0.3% representation. Census works equally well with both unrelated individuals as well as families, including trios as shown below.

From the manuscript, figures on census accuracy.



A. Results of *in silico* data-mixing experiments to estimate the bias and variance of Census-seq inferences for donors who have contributed small proportions (0.05% to 1%) to a mixture. WGS data from 40 unrelated donors was mixed *in silico*, with 30 donors at an arithmetic series of representations from 0.05% to 1.00% in 0.05% increments, and 10 donors (for whom data not shown) at higher representations, such that the 40 donors' representations summed to 1. This was repeated for 10 simulations, in each of which donors were permuted between the low and high representation groups at each iteration to generate a total of 300 observations of each bin.

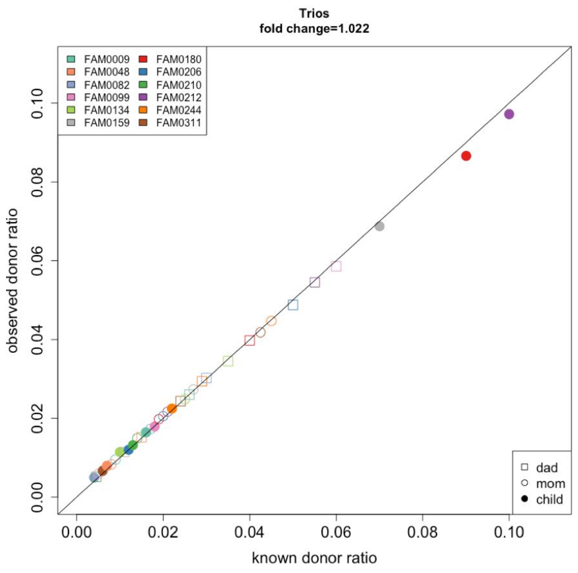
B. Bias of Census-seq estimates (as a fraction of the estimate) for donors who have contributed small fractions (< 1%) of a cell or DNA mixture. Using the data shown in S1F, the median absolute error in representation was calculated as exp(median (abs(log(donor % representation in silico / donor % representation inferred by Census-seq)))). Bias in donor representation was substantially greater for donors contributing <0.3% to a mixture. Bias was <15% at a representation of 0.3%, and <10% at a representation of 0.4%. We believe that this bias arises from PCR and sequencing errors, which result in reads that (as a group) tend to bias upward the estimated representation of very-low-contribution donors. For this reason, we exclude from some genetic analyses those donors with contributions of <0.3% to a mixture.

C. To assess the potential effect of having genetically related donors in a mixture, three *in silico* mixtures of 40 donors were constructed for: 40 unrelated individuals; 20 parent/child pairs; and 20 sibling pairs. In each analysis, WGS data from these donors was mixed uniformly at a representation of 0.025; the data mixture was then analyzed by Census-seq. Error in representation was calculated as the difference between the known and Census-seq-inferred donor-contribution estimates. 95% of inference were within an absolute error of 0.001. The median absolute error in representation estimates were similar: unrelated 3.4×10^{-4}, sibling= 4.1×10^{-4}, parent-child= 2.6×10^{-4}.

D. To evaluate the robustness of Census-seq inference to the inclusion of genetically related individuals in a village, *in silico* data mixing was used to simulate a village of 20 sibling pairs, with the same distribution of representations as in Fig. 1E.

E. To evaluate the robustness of Census-seq inference to the inclusion of genetically related individuals in a village, *in silico* data mixing was used to simulate a village of 20 parent-child pairs, with the same distribution of representations as in Fig. 1E.

Not included in the paper (due to inability to share raw data at time of publication), we also tested the accuracy to ascertain donor representation in trios. Here, a pool of 36 related donors [12 sets of trios] are mixed in known proportions in silico, to ~ 2x total depth, then analysed by Census-seq.



Invoking CensusSeq

```
CensusSeq
INPUT_BAM=my.bam
INPUT_VCF=my.vcf.gz
SAMPLE_FILE=my.samples.txt
O=my.census.txt
IGNORED_CHROMOSOMES=null IGNORED_CHROMOSOMES=chrX
IGNORED_CHROMOSOMES=chrY IGNORED_CHROMOSOMES=chrM
```

Depending on the version of the reference genome used, you may wish to set a different name for the sex and MT chromosomes. In the example above, we've taken the standard contig names expected (X,Y.MT) and replaced them with chrX, chrY, and chrM. Depending what genome you aligned to, you may or may not have to set these values, but we highly recommend you only run Census-seq using the autosomes.

Example output:

```
#INPUT_BAM=parent-child_varying.bam
INPUT_VCF=CIRMw1w2w3.no_twins_clean.vcf.gz    GQ_THRESHOLD=30
FRACTION_SAMPLES_PASSING=0.9   MIN_NUM_VARIANT_SAMPLES=2    READ_MQ=10
#NUM_POSSIBLE_SNPS=8511425    NUM_SNPS_USED=7742325    CONVERGED=true
BEST_LIKELIHOOD=-3298641.931987352     SECOND_LIKELIHOOD=-3298641.994792817
LIKELIHOOD_DELTA=0.06280546495690942 NORMALIZED_LIKELIHOOD=-0.1635309551986775
8    SHANNON_WEAVER_DIVERSITY=3.54     SHANNON_WEAVER_EQUITABILITY=0.96
DONOR                          REPRESENTATION
60054GG1_P13_MT_4.4.18       0.04394
CW60014_P14_DH_11-10-17     0.00332
CW60018_P12_DH_11-14-17     0.02621
CW60026_P13_DH_11-13-17     0.0211
CW60027_P13_MT_11-16-17     0.01719
CW60055_P13_MT_12-20-17     0.0103
```

As you can see above, there are informational header lines to remind you of what your program arguments were when this data was generated on the first line. The second informational header contains some metrics about the run:

1) The number of SNPs that were valid in the VCF and could be used in analysis
2) The number of SNPs that also had at least one informative read, and were used in the analysis
3) If the algorithm converged. If this is false, you'll need to "debug" your pool and VCF to determine where the problem is, but the results should not be trusted.
4) The likelihood of the data given the mixture.
5) The likelihood of the data given the mixture on the penultimate iteration.
6) The likelihood of the data per SNP.
7) Diversity and equitability measures of the data - how close to the uniform distribution are the donors?

These headers are prefixed by "#", which is very convenient if you use the R programming language. The main output contains the donor label and the representation of that donor in the pool. The total representation sums to 1 across all donors.

Census-seq problems you may encounter

Number of total SNPs low - You should check the quality control of your VCF. In particular, the number of variants that have low call rates across many individuals, as well as the genotype quality of those cells.

Number of SNPs used is very low but the number of total SNPs is high - Problems with sequencing quality. When the number of SNPs is low, the algorithm may not converge, or the results have higher variance than desired.

Algorithm did not converge - This can be a data quality problem, but is most frequently seen when the true set of donors in the pool differ greatly from the set of donor genotypes that are optimized. Try running RollCall on the data set to see if there's been a sample swap.

Validating CensusSeq with Synthetic Data

If you wish to repeat the same validation analysis we performed for the paper, there are just a few steps to generate your own synthetic data set. To follow this analysis, you'll need a set of BAM files that each contain sequence data for a single donor.

Tag each BAM file with the name of the donor of origin of the reads.

If you want to take full advantage of the built-in validation, you should use the same identifier as found in the VCF. The dropseq software package includes a program to do this.

```
TagBam I=my.bam O=my.tagged.bam TAG_NAME=ZS TAG_VALUE
```

Merge all BAM files together into a single file using Picard's MergeSamFiles.

```
MergeSamFiles I=my.tagged1.bam I=my.tagged2.bam O=my.final_tagged.bam
```

Downsample BAM so that donors are at some user-defined set of proportions

```
DownsampleBamByTag
I=my.final_tagged.bam
TAG=ZS TAG_FILE=donor_counts.txt
O=synthetic_data.bam
```

The TAG_FILE contains 2 columns, the donor ID and the number of reads to select at random from the source BAM, with no header. This downsampling is probabilistic by default, so the output will be very close to the exact number, but may vary by a very small fraction.

Example donor_counts.txt:

```
CW20105_P14_DH_12-8-17    132352
CW60045_P12_MT_11-9-17    132352
CW60079DD1_P13_MT_4.6.18    236842
```

Run Validation

Invoke CensusSeq on the newly generated synthetic data with the appropriate VCF, and add the argument KNOWN_DONOR_TAG=ZS. This will add 2 additional columns to the output:

```
DONOR    REPRESENTATION    KNOWN    COUNT
```

```
CW20073_P13_MT_12-8-17    0.03099    0.03083    1387689
CW20094_P14_MT_12-8-17    0.04511    0.04476    2014675
CW20105_P14_DH_12-8-17    0.00314    0.00293    131994
CW20183_P13_MT_12-16-17    0.04031    0.04007    1803702
```

The COUNT column is the absolute number of reads observed that pass filtering thresholds for the program. We emit the filtered counts instead of the unfiltered counts because different BAM files are different experimental batches. For our data sets, each BAM file has a different proportion of reads that map, which if not controlled for appears to be a mistake in the Census-seq algorithm, but is in fact a difference in quality of the input data. In a true experiment, this batch effect should not occur as all donors are sequenced at the same time. The KNOWN column is the proportion of READS / sum(all READS).

You can then plot the KNOWN and REPRESENTATION columns against each other to reproduce the synthetic validation plots.

These options are also available for RollCall and CsiAnalysis.

Roll Call

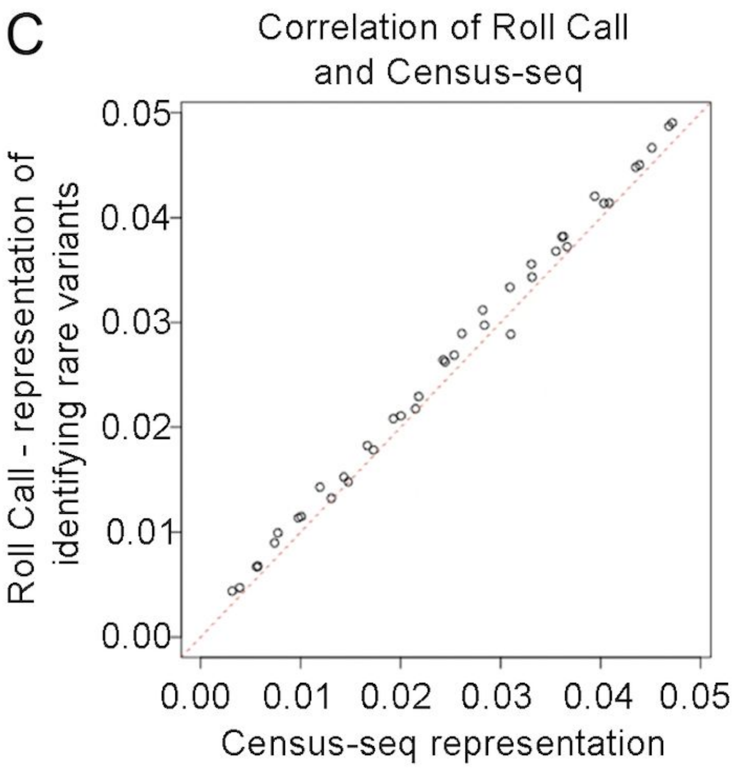
In any population-scale project, sample swaps can occur even despite best efforts. We developed an algorithm (“Roll call”) to catch such swaps quickly and automatically. We use each individual’s private (singleton) alleles to find evidence that an unexpected (but genomically “known”) donor from the project is present in a pool in which s/he doesn’t belong.

To do this, we measure the fraction of the sequence data that contains rare variants private to each individual. Since these IRVs (**I**dentifying **R**are **V**ariants) are the only source of alternate alleles in the sites observed, the fraction of these alleles observed in a pooled sequencing experiment can be directly related to the proportion of the donors in the pool. We count the observations (sequence reads) that (in the absence of sequence errors) could only arise from a given individual, divided by the total number of reads at those sites. This is less accurate than Census-seq at quantifying donor representation, but can search a very large set of potential donors to determine presence or absence.

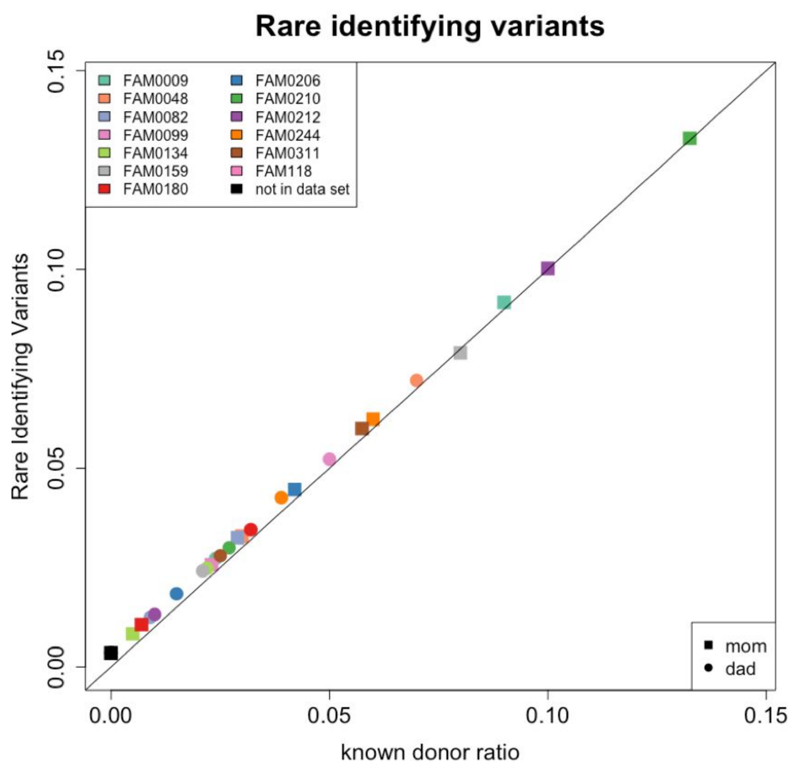
The advantage of Roll Call is that it’s not restricted to any sample list - this is a simple “counting” algorithm without an optimization step, so it’s straightforward to learn which set of donors are in your experiment without prior knowledge. This is especially useful to validate Census-seq results, as the two results are highly correlated. Donors that are not expected in the pool have a non-zero value due to sequencing and genotype errors, but these donors have much lower representation than the donors in the pool.

Relying on private SNPs alone to quantify the representation of donors does provide some challenges. Pools with related individuals will effectively have fewer SNPs to ascertain, as closely related individuals will share SNPs. In the case of trios, ascertainment of children becomes impossible, as they don't have variation (aside from a handful of denovo SNPs) to differentiate children from their parent genotypes. Using only private SNPs also restricts the total number of donors that can be ascertained in a pool, as adding additional donors restricts the set of SNPs that will be private to any one donor - in a smaller pool low frequency variants may appear private as they are only observed in a single individual, but as population size increases many of those alleles will appear by chance in another individual in the population.

Below is figure S2C from the CensusSeq paper, demonstrating the correlation between CensusSeq and RollCall.



Below is an example of roll call output, again on the same set of trios as the census example, but children are excluded from the output:



Invoking Roll Call

```
RollCall
INPUT_BAM=my.bam
INPUT_VCF=my.vcf.gz
O=my.roll_call.txt
IGNORED_CHROMOSOMES=null IGNORED_CHROMOSOMES=chrX
IGNORED_CHROMOSOMES=chrY IGNORED_CHROMOSOMES=chrM
```

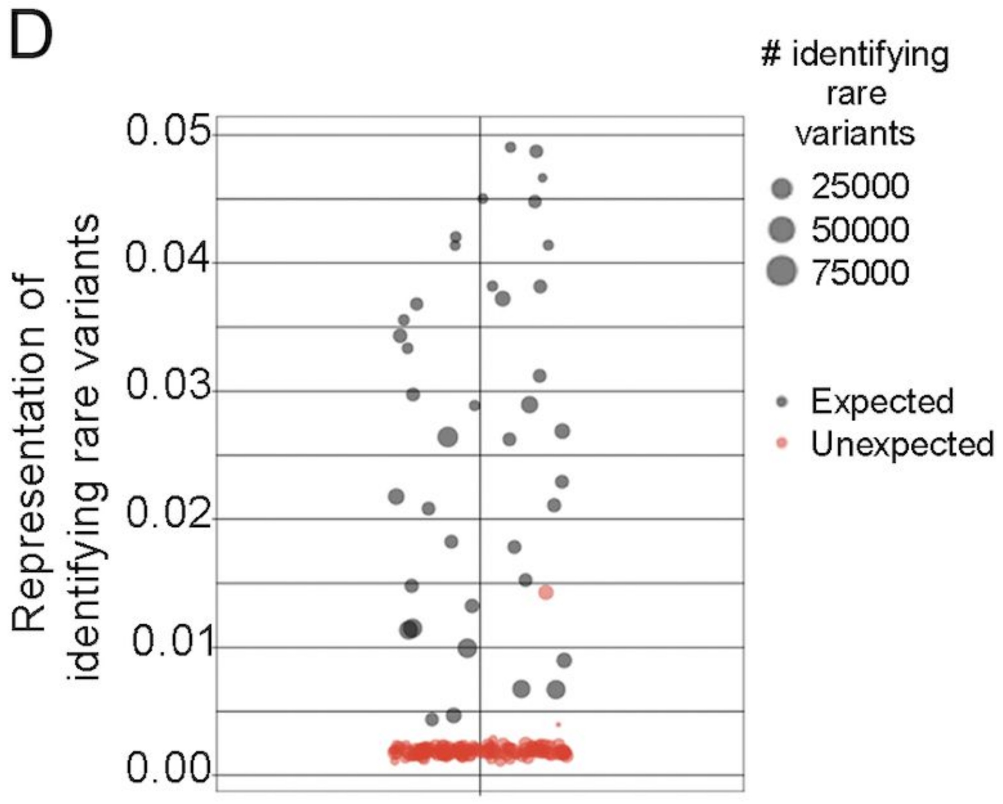
See CensusSeq documentation for our advice on the best settings for the IGNORED_CHROMOSOMES option.

Example Output

```
#INPUT_BAM=unrelated_varying.bam    INPUT_VCF=CIRMw1w2w3.no_twins_clean.bcf
GQ_THRESHOLD=30    FRACTION_SAMPLES_PASSING=0.9    MIN_NUM_VARIANT_SAMPLES=1
READ_MQ=10
#NUM_POSSIBLE_SNPS=5531902    NUM_SNPS_USED=5034777
DONOR    REPRESENTATION    REP_IRVs    NUM_SNPS
60054GG1_P13_MT_4.4.18    0.00148    0.00197    40699
CW20012_P14_MT_12-17-17    0.00113    0.00151    5029
CW20025_P13_MT_11-22-17    0.00136    0.00182    24526
CW20031_P12_DH_11-24-17    0.00139    0.00185    25821
```

See the CensusSeq documentation for information in the header.

The main output contains 4 columns by default. DONOR is the donor identifier. REPRESENTATION is the proportion of the donor in the pool, where the sum of all representations is 1. REP_IRVs is the proportion of alternate alleles / alt + ref alleles at SNPs where the donor is heterozygous and all other donors in the pool are homozygous (private SNPs). REP_IRVs is not normalized to sum to one. NUM_SNPS is the total number of variants that were heterozygous for this donor, but homozygous for all other donors in the set of tested donors, and had read coverage in the BAM file.



The above is figure S2D from the supplemental. Here, we're using a VCF file with 187 donors to determine which donors are in the pool. The Y axis plots the REP_IRV column from the output, and each point's size is determined by the NUM_SNPs column, while the red and black colors are based on our expectation of which donors we expect should be in the pool.

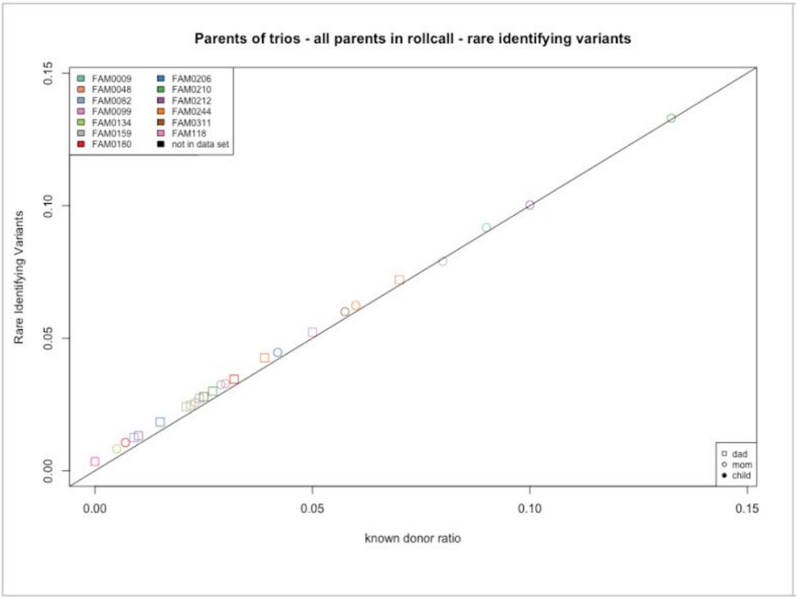
Because of sequencing and genotype errors, the number of alternate alleles seen for donors not in the pool across many thousands of sites is rarely 0, but the distribution of REP_IRV for donors we expect to be near 0 is generally in the range of 0 to 0.005. Donors that are related to many individuals may have many fewer SNPs to ascertain their REP_IRV, so have a higher variance in that score. In this case, a donor with very few SNPs (1755) appears to have a score higher than the bulk of the known not present donors, but we can safely assume that the donor is likely not present in the pool, but the slightly elevated score above the null is due to this ascertainment issue. If this donor had a much larger set of SNPs (and thus less variance in their ascertainment), we would consider the donor to be present due to contamination or a sample swap.

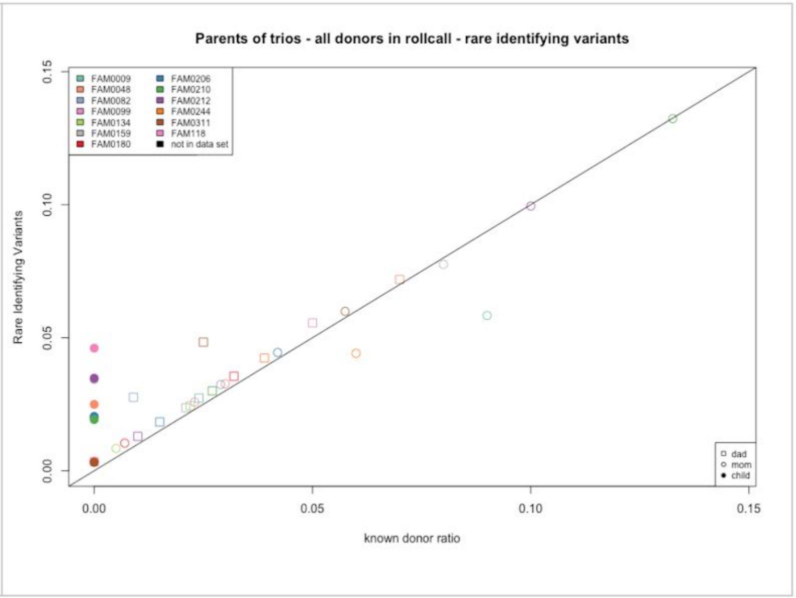
The red donor at Y=0.146 is a better example of a potential sample swap. In this case, the donor is well separated from the null distribution, and has many more SNPs (>35000).

Roll Call with Trios
Roll call is typically run on all donors available in the VCF file to identify potential donor swaps, so no sample file is required to run the analysis.

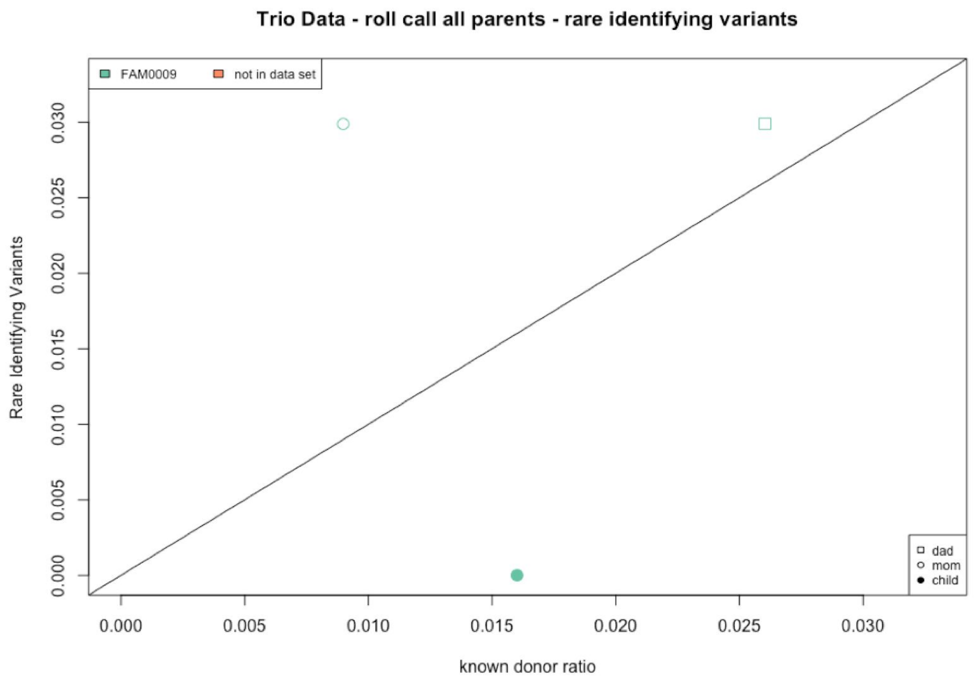
Rare identifying variants typically give a very similar result to census when looking at the correct set of donors. However, if assessing trio data, it's important to use the sample list parameter and only include the parents in the list. Due to the fact that children share the genotypes of their parents and have very few private SNPs, they are not well estimated.

In the panels below, the donors analyzed are the parents of trios. When roll call is run with a sample list including only the parents of trios, the results on the left are generated. When roll call is run to include both the parents and children, in the results on the right the children are mis-estimated. There are almost no private SNPs for the children, so a single sequence error can greatly increase the REP_IRV score.





Thus, roll call can identify sample swaps of parents, but is unable to ascertain children directly. However, if there is a sample swap event, a child can be detected by way of the parents. Below is a single trio's REP_IRV values.



In this case, roll call can not see the child directly (the IRV score is low compared to the known amount of the donor), but the alleles of the child are shared by the parents, so the parents REP_IRVs are higher than their known estimates. Roll call is observing each parent + some share of the child's contribution to the data.

This enables you to infer that a sample swap occurred and a donor from a specific family was involved, but you aren't guaranteed to identify the donor. If the donor was one of the parents, roll call will only observe the one parent. If the swapped in donor was a child, then you see a signal from both parents, but you can't rule out that the entire trio was swapped in.

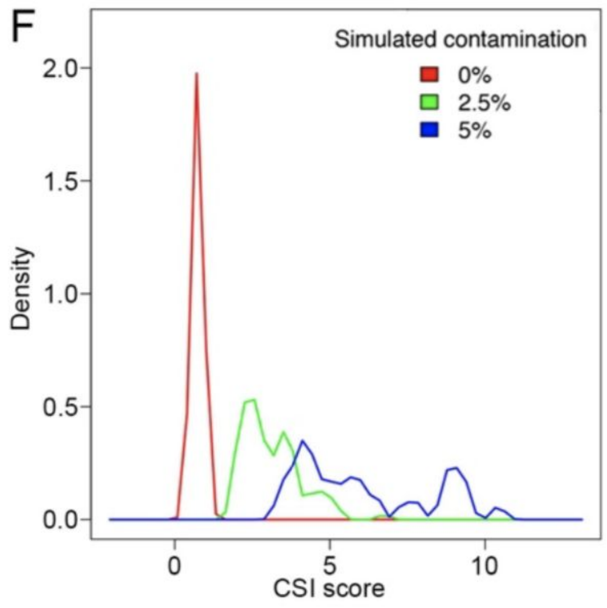
CSI (Contaminating Sample Identifier)

What if a pool were visited by cells from a genomically "unknown" donor? If we don't have genotypes for the contaminating donor(s) *a priori*, we need another way to detect these "unknown unexpected" visitors. We developed the CSI algorithm to do this.

CSI utilizes observations of alternate alleles that could not have arisen from the expected donors' genomes and must therefore represent sequencing errors or contaminating cells. We first identify all alleles that are absent among the donors we expect in the pool but present at some minimum frequency in the wider population as estimated from the Thousand Genomes Project or gnomAD data, or the population of donors in the VCF.

By correcting for sequencing error rate, we distinguish between two models: sequencing errors and an unwelcome visitor.

CSI is by far the hardest of the algorithms to calibrate and interpret, due to the number of unknown factors for the unknown donor(s), which can lead to underestimates of the contamination level. The base case would be an individual who is unrelated to anyone in your pool, and is from a similar population (with similar allele frequencies) to the population you used to estimate those frequencies. In the base case, the estimates of contamination are fairly accurate [see figure below], but it's important to understand how the algorithm can be less accurate if the contaminating donor is related to someone in the pool, or is from a different population.

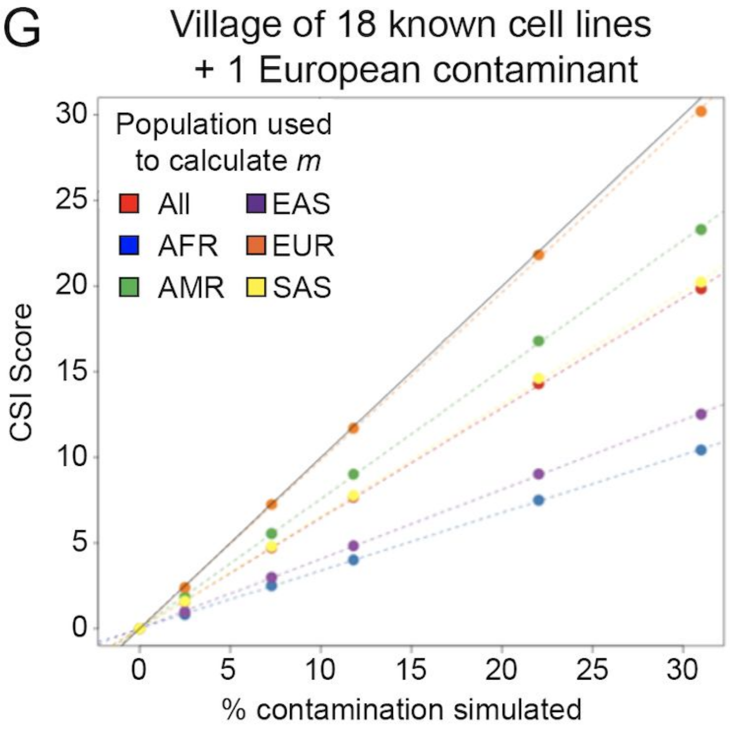


Distributions of CSI scores for *in silico* villages created by WGS data mixing to have 0%, 2.5%, or 5% contamination from an "unknown" donor to whose genetic data the CSI analysis was blinded. The *in silico* mixing experiments each involved a mixture of 39 known, unrelated donors at varying concentrations and (in the 2.5% and 5% cases) an additional randomly selected unrelated donor, to whose genetic data the CSI analysis was blinded. 100 simulated villages per contamination level were analyzed. The 100 null (uncontaminated) simulations yielded CSI scores of 0.0074 +/- 0.0011; CSI scores in all 100 of the 2.5% contamination simulations exceeded any result from this null distribution.

If the unknown contaminating donor is related to an individual already in the pool, then they will share some fraction of their alleles, and those alleles will be attributed to the known donor in the pool. Thus, a first degree relative (parent/child, sibling) is the contaminant, they share ~50% of their alleles, and the estimate of contamination will be about half of the true value.

If the unknown donor is from a different population, then CSI will underestimate the contamination rate. Below is figure S2G from the paper.

G



To understand this figure, it's important to think about how the CSI algorithm works. The variants that are ascertained by CSI are selected by 3 inclusion criteria:

- The variant's frequency in the reference population is "common", that is, it has an MAF > 2.5%
- The variant has no copies of the alternate allele in the genotypes of the donors that are "known" to be in the population
- The variant has some amount of sequencing data at the location

When we use a reference population that is different from the population the unknown donor was drawn from, why are these rules important?

All SNPs that are common in the "wrong" reference population but absent in the population the donor belongs to would be included - other donors from the pool would not filter out these sites. Additionally, you'd expect to see no examples of the alternate allele in the individual. When we calculate the frequency of sequencing reads at these sites that contain the minor allele, these sites would contribute to the denominator (observed reference) but not the numerator, which would decrease our estimate of the frequency of alternate alleles at these sites.

This would similarly omit some informative sites that are absent in our population that are common in the "true" reference population, but are absent in the "wrong" reference population.

We would expect to see more % contamination than expected at sites where the variant was common in both populations, but was more common in the "correct" reference population. Here, the result is "more surprising" because you see more alternate alleles than you'd expect, but those sites would likely be filtered by everyone else in the pool.

Invoking CSI

```
CsiAnalysis
INPUT_BAM=my.bam
INPUT_VCF=my.vcf.gz
SAMPLE_FILE=my.samples.txt
O=my.csi.txt
IGNORED_CHROMOSOMES=null IGNORED_CHROMOSOMES=chrX
IGNORED_CHROMOSOMES=chrY IGNORED_CHROMOSOMES=chrM
```

Like CensusSeq, CSI relies on a list of donor IDs that are expected to be in the pool. This set of donors in turn defines the set of variants that are expected to be homozygous for all individuals in the pool, and will be ascertained in the sequencing data.

By default, CsiAnalysis calculates the allele frequency of each variant from all donors in the VCF that are not in the donor pool. If you wish to calculate contamination rates using a different reference population, you can use the ALLELE_FREQ_TAG argument to specify a tag name on records of the VCF file that contains the allele frequency of each variant in the population you want to test (see: Add reference population allele frequencies.)

Example Output

```
#INPUT_BAM=unrelated_varying_CSI_calibration_5_99.bam
INPUT_VCF=CIRMw1w2w3.no_twins_clean.bcf
SAMPLE_FILE=unrelated_varying_CSI_calibration_5_99.donor_list MINIMUM_MAF=0.025
READ_MQ=10    MIN_BASE_QUALITY=10
## METRICS CLASS    org.broadinstitute.dropseqrna.censusseq.CsiAnalysis
NUM_SNPS REF_COUNT ALT_COUNT MEAN_MAF FRAC_ALT SEQUENCING_ERROR_RATE
73152    192744    573       0.034189 0.002964         0.001
PCT_CONTAMINATION
5.744736
```

As you can see above, there are informational header lines to remind you of what your program arguments were when this data was generated on the first line.

Output columns

```
NUM_SNPS - the number of variants with a MAF > MINIMUM_MAF in the VCF where all
donors in the pool were homozygous reference with at least one sequence read in
observed in the BAM
REF_COUNT - for the SNPs observed, how many reads observed the reference allele
ALT_COUNT - for the SNPs observed, how many reads observed the alternate allele
MEAN_MAF - The average minor allele frequency of the SNPs selected
FRAC_ALT - ALT_COUNT/(ALT_COUNT+REF_COUNT)
SEQUENCING_ERROR_RATE - The sequencing error rate default estimate
PCT_CONTAMINATION - Result from plugging the above numbers into the CSIAnalysis
formula
```

Putting it all together - SOP for CensusSeq

For any CensusSeq run, we suggest the following set of steps to perform QC:

All donors present

First, run all 3 CensusSeq programs on your aligned BAM and VCF. Run CensusSeq and CsiAnalysis with the SAMPLE_FILE argument pointing to the expected list of donors, and run RollCall on the entire VCF. If things went well, then you should observe the following:

1) The CensusSeq algorithm converges
2) Roll Call detects the same set of donors as CensusSeq, and no additional donors are detected above the null distribution
3) CsiAnalysis % contamination rate is <=2%.

If you meet these criteria, you're finished with your QC and should be able to use your results with downstream analysis.

Sample Swap

However, we're sometimes not so lucky, and need to "debug" the mixture of donors in the pool. Here's an alternative scenario, where there's a sample swap in the pool where the unexpected donor is also genotyped, so you can identify exactly who was swapped.

1) The CensusSeq algorithm may or may not converge - for a single sample swap, CensusSeq will be robust to this change. For a large number of swaps in a single pool, CensusSeq may not be able to explain this incorrect set of donors.
2) Roll Call detects a new donor that was unexpected in the original donor list. Note that one or more donors that are expected may not be detected - if multiple donors are at very low representation, it may be hard to say which of those donors was swapped for the new unexpected but known donor.

3) CsiAnalysis % contamination rate will be approximately the representation of the known unexpected donor.

When this occurs, we find it best to generate a new sample list that contains the new unexpected donor, and rerun all 3 programs. When a sample swap is detected and “fixed” in this fashion, the second run of CensusSeq programs should return results similar to when you submit a pool with the proper donor list.

Contamination

Another scenario you may encounter is a contamination event. In this scenario, the pool has one or more additional unexpected, unknown donors that we don’t have genotypes. In this case, you’ll see the following pattern:

1) The CensusSeq algorithm may or may not converge - for a contamination event, CensusSeq may be robust to this change. For a large proportion of contamination in a single pool, CensusSeq may not be able to explain the data. Alternatively, census can uniformly distribute the unexpected data across the known donors as noise, which will cause the entire population to shift towards a uniform distribution. We first observed this effect with a pool that started off with donors having very different proportions in the pool, and gradually shifting towards a uniform distribution. The data converging towards a perfectly balanced pool was “too good to be true”, and motivated us to write CsiAnalysis.
2) Roll Call will not detect new individuals.
3) CsiAnalysis % contamination rate will be approximately the representation of the unknown unexpected donor. You should be concerned that you have contamination when you see a CsiAnalysis score >=2%. If the contaminating donor(s) have higher growth rates than the known donors in the pool, contamination rate should increase at subsequent passages.