Variant Calling with R/Bioconductor

Michael Lawrence

December 14, 2016

Outline

Outline

Outline

Goals and Scope

- Determine the genotype of a sample
- Call single nucleotide variants vs. reference from high-throughput sequencing data, including WGS, Exome-seq and (eventually) RNA-seq
- Support users to filter the variant calls according to the biological context and questions of interest
- Be sensitive to low frequency variants
  - Be robust to aneuploidy, cell mixtures, contamination
  - Permit estimation of sample heterogeneity

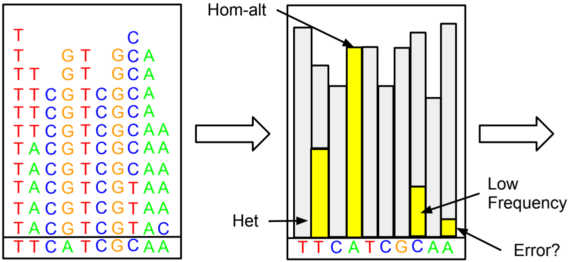
Variant Calling Process

Data Generation

1. Library prep (PCR)
2. Sequencing
3. Alignment

Each of these steps will introduce noise that requires filtering.

Variant Calling



POS	REF	ALT
2	T	A
4	A	G
8	C	T

Biological Considerations

These generate a range of variant frequencies:

- Aneuploidy
- Heterogeneity
- Contamination

Thus, *there is no "one-p-fits-all" solution to variant calling*.

Existing Solutions

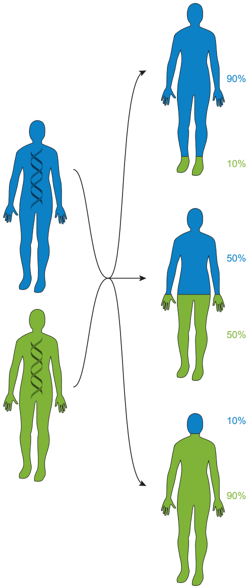
Other tools for calling variants vs. reference include:

`samtools mpileup`	Generates statitics useful for variant calling
`vcfutils`	Perl script for filtering `mpileup` output
`Varscan2`	Series of adhoc filters on `mpileup` output
`GATK`	Oriented towards genotyping in diploid samples

There are also comparative (somatic mutation) callers (strelka, MuTect, etc), but we are focused on calling vs. reference.

Benchmark Dataset

- To develop an algorithm, we need to benchmark its sensitivity and specificity, but no gold standard exists.
- Biochemically mixed two HapMap daughter cell lines in different proportions to realistically simulate variant frequencies expected from complex samples. Sequenced each genome with 75bp reads.



Sequencing Output: 23-24X average coverage

Sample	% CEU	% YRI	# Reads (analyzed)	Avg. Coverage
1	90	10	461,449,560	22.3
2	90	10	475,567,437	23.0
3	90	10	460,196,498	22.3
4	50	50	489,166,262	23.7
5	50	50	442,737,941	21.4
6	50	50	430,779,023	20.8
7	10	90	496,958,600	24.0
8	10	90	494,245,570	23.9
9	10	90	534,458,340	25.8

Genotypes

Cell Line	Trio	Source	Ref	Coverage	Total Het/Hom
NA12878	CEU	Broad	hg19	64X	2451814/1410358
NA12878	CEU	1000G	hg18	61X	1703706/1061942
CEU Union	CEU	Both			2424095/1427209
NA19240	YRI	1000G	hg18	66X	2227251/1108784

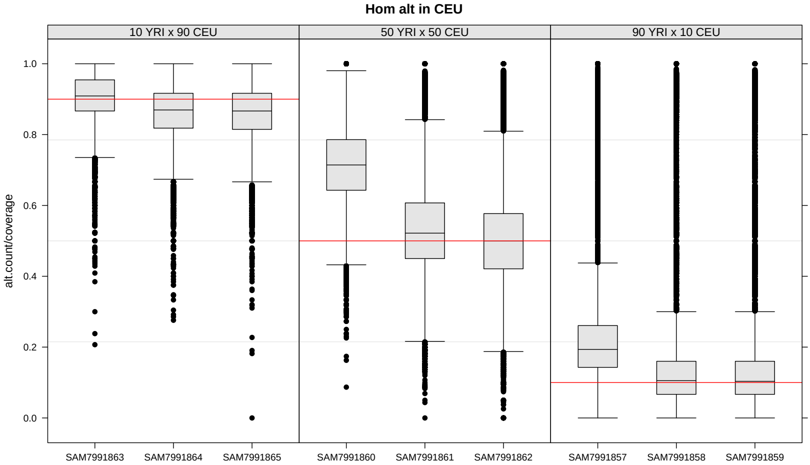
10/90 combinations

10/90	0	0.5	1
0	-	0.45	0.90
0.5	0.05	0.50	0.95
1	0.10	0.55	1.0

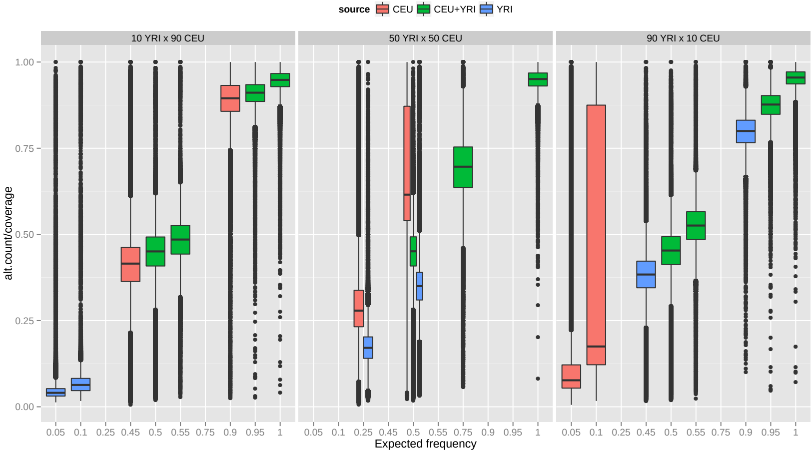
50/50 combinations

50/50	0	0.5	1
0	-	0.25	0.50
0.5	0.25	0.50	0.75
1	0.50	0.75	1.0

QC of mixture ratios

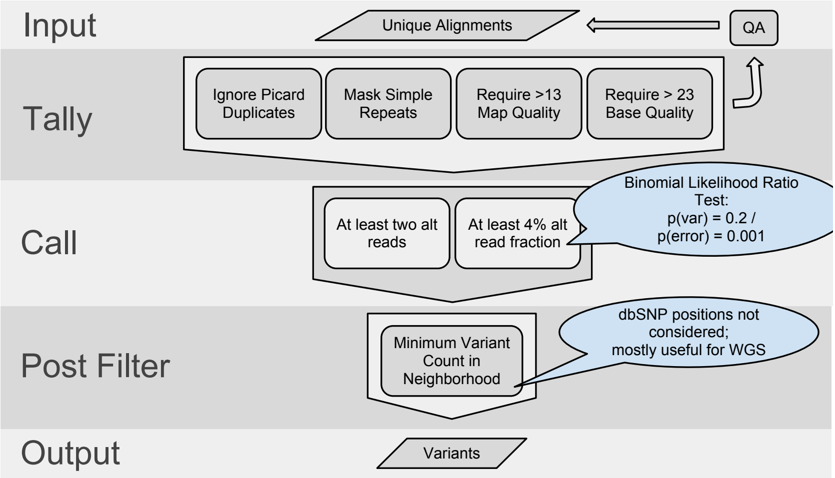


QC of variant frequencies



Outline

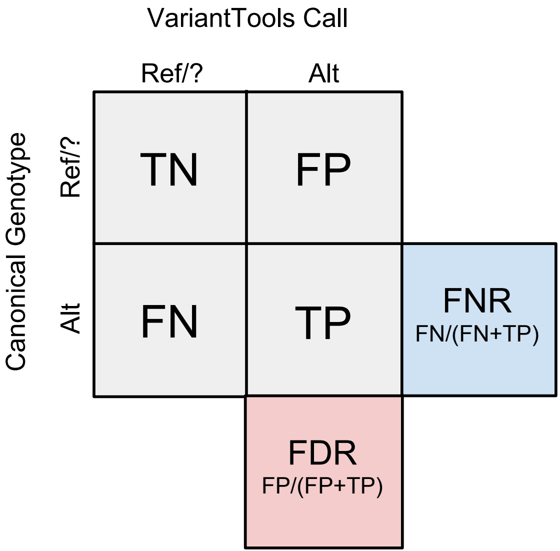
Overview


Input
Unique Alignments
QA
Tally
Ignore Picard Duplicates
Mask Simple Repeats
Require >13 Map Quality
Require > 23 Base Quality
Call
At least two alt reads
At least 4% alt read fraction
Binomial Likelihood Ratio Test:
p(var) = 0.2 /
p(error) = 0.001
Post Filter
Minimum Variant Count in Neighborhood
dbSNP positions not considered;
mostly useful for WGS
Output
Variants

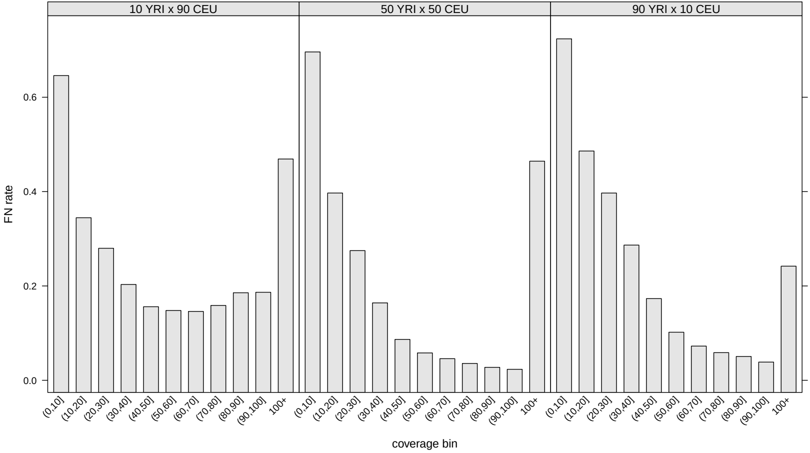
Outline

- Introduction to the dataset
  - Experiment
  - Algorithm
  - Performance
- Interactive demonstration
  - Overview
  - Alignment
  - Variant calling
  - Exploratory analysis

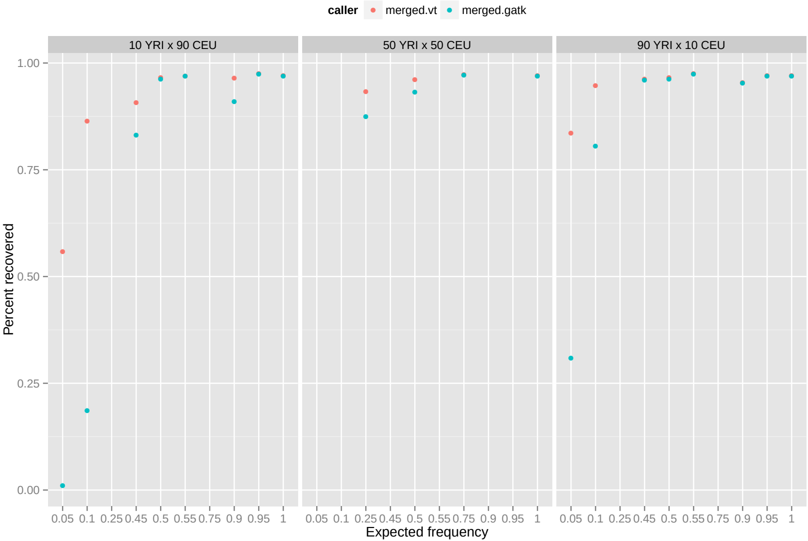
Definitions



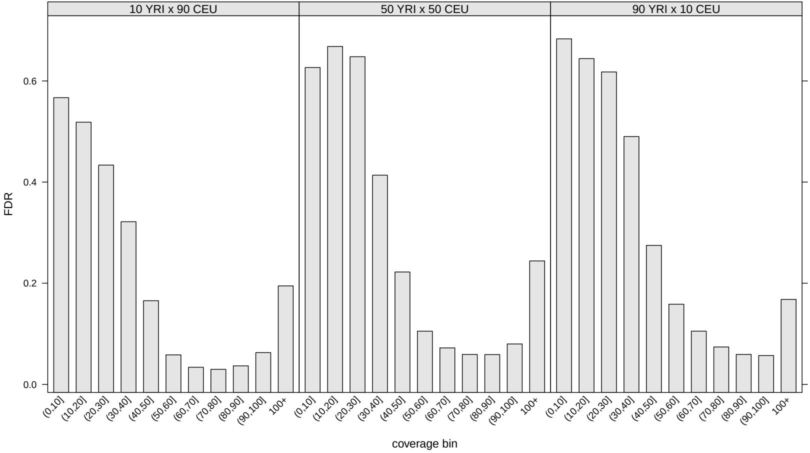
FNR high at low/high coverage



Recovery rate (1 - FNR) vs. GATK

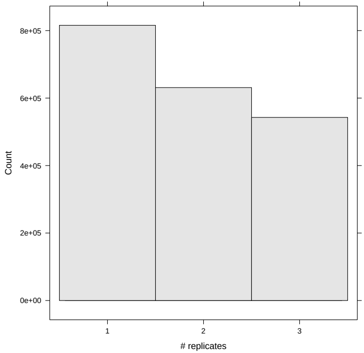


FDR by coverage bin



Evidence that some FP are real

Replication

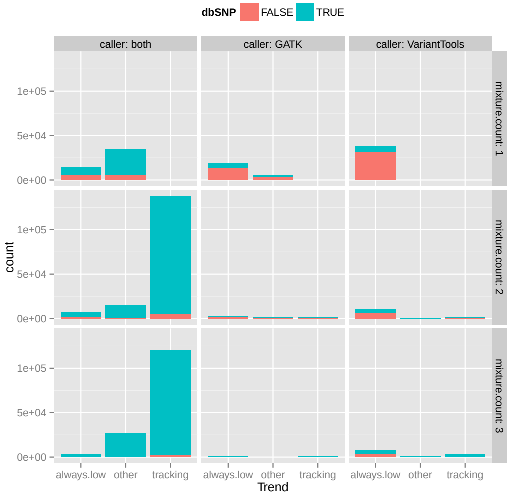


dbSNP Concordance

	NOT dbSNP	IN dbSNP
1 Rep	695468	120266
2+ Reps	391879	781940

Selected FP: GATK vs. VariantTools

Selected FPs at reasonable (45-85X) coverage, outside of structural variants and multi-mapping regions.



Acknowledgements

Leonard Goldstein
Melanie Huntley
Yi Cao
Robert Gentleman

Outline

Outline

Overview

Data

Subset of the mixture data consisting only of the 50/50 samples, and only reads aligning within 1 Mb of p53.

Strategy

1. Align sequences to the p53 region.
2. Generate tallies (pileup) from the alignments.
3. Call/filter variants.
4. Perform exploratory analysis on the calls and concordance with canonical genotypes.

Outline

The gmapR package

gmapR is an R interface to the GMAP/GSTRUCT suite of alignment tools, including:

- **GSNAP** a short read aligner distinguished by its ability to generate spliced alignments from RNA-seq data (also handles DNA)
- **`bam_tally`** summarizes alignments by counting A/C/G/T (and optionally indels) at each position and tabulating by strand, read position and quality

Configure GSNAP parameters

- GSNAP is a complex tool with a complex interface, consisting of many command-line parameters.
- gmapR supports all parameters, while providing a high-level interface with reasonable defaults.
- The parameters are stored in a GsnapParams object.
- We construct a simple GsnapParams for generating unique DNA alignments to ~2Mb region around p53:

```
library(gmapR)
param <- GsnapParam(TP53Genome(), unique_only = TRUE,
                    molecule = "DNA")
```

Align with GSNAP

We find our FASTQ files inside the VariantToolsTutorial package:

```
extdata.dir <- system.file("extdata",
                           package="VariantToolsData")
first.fastq <- dir(extdata.dir, "first.fastq",
                   full.names=TRUE)
last.fastq <- dir(extdata.dir, "last.fastq",
                  full.names=TRUE)
```

And generate the GSNAP alignments (for the first sample), which gmapR automatically converts to indexed BAMs:

```
output <- gsnap(first.fastq[1], last.fastq[1], param)
bam <- as(output, "BamFile")
```

Outline

The VariantTools package

VariantTools is a set of utilities for:

- Tallying alignments (via gmapR)
- Annotating tallies
- Filtering tallies into variant calls
- Exporting tallies to VCF (actually VariantAnnotation)
- Wildtype calling (for a specific set of filters)
- Sample ID verification via rudimentary genotyping

Generate nucleotide tallies

The underlying bam_tally from GSTRUCT accepts a number of parameters, which we specify as a TallyVariantsParam object. The genome is required; we also mask out the repeats.

```
library(VariantTools)
data(repeats, package = "VariantToolsData")
genome(repeats) <- genome(TP53Genome())
param <- TallyVariantsParam(TP53Genome(), mask = repeats)
```

Tallies are generated via the tallyVariants function:

```
tallies <- tallyVariants(bam, param)
```

Loading and combining three samples worth of tallies

The alignments and tallies were generated for all three replicates of the 50/50 mixture and placed in the package.

```
data(tallies, package = "VariantToolsData")
```

We combine the samples in two different ways: stacked (long form) and merged (depths summed).

```
stacked.tallies <- stackSamples(tallies)
merged.tallies <- sumDepths(tallies)
sampleNames(merged.tallies) <- "merged"
```

Configure filters

VariantTools implements its filters within the `FilterRules` framework from IRanges. The default variant calling filters are constructed by `VariantCallingFilters`:

```
calling.filters <- VariantCallingFilters()
```

Post-filters are filters that attempt to remove anomalies from the called variants:

```
post.filters <- VariantPostFilters()
```

Filter tallies into variant calls

The filters are then passed to the `callVariants` function:

```
merged.variants <- callVariants(merged.tallies,
                                calling.filters,
                                post.filters)
```

Or more simply in this case:

```
merged.variants <- callVariants(merged.tallies)
stacked.variants <- callVariants(stacked.tallies)
```

Or, call variants directly from a BAM

```
variants <- callVariants(bam, param)
```

Note

Convenient for simple exercises, but does not facilitate diagnostics

Outline

Alternative allele frequencies

Check the quality of our mixtures:

```
stacked.variants$altFraction <-
  altDepth(stacked.variants) / totalDepth(stacked.variants)
library(ggplot2)
qplot(altFraction, geom = "density", color = sampleNames,
      data = as.data.frame(stacked.variants))
```

Annotating variants with genotype concordance

We want to see how well our calls recapitulate the genotypes from 1000G; we have these prepared as a dataset:

```
data(geno, package = "VariantToolsData")
```

Merge the expected frequencies of each alt with the variant calls:

```
naToZero <- function(x) ifelse(is.na(x), 0L, x)
addExpectedFreqs <- function(x) {
  expected.freq <- geno$expected.freq[match(x, geno)]
  x$expected.freq <- naToZero(expected.freq)
  x
}
stacked.variants <- addExpectedFreqs(stacked.variants)
merged.variants <- addExpectedFreqs(merged.variants)
```

Annotating the genotypes with merged variant calls

Annotate the genotypes for whether an alt allele was called in the merged data, and also add the alt and total depth:

```
softFilterMatrix(geno) <-
  cbind(in.merged = geno %in% merged.variants)
mean(called(geno))
```

```
0.710044395116537
```

```
m <- match(geno, merged.tallies)
altDepth(geno) <- naToZero(altDepth(merged.tallies)[m])
totalDepth(geno) <- naToZero(totalDepth(merged.tallies)[m])
```

False negatives: which filter to blame?

Apply the calling filters to our FN and summarize the results:

```
fn.geno <- geno[!called(geno)]
fn.geno <- resetFilter(fn.geno)
filters <- hardFilters(merged.variants)[3:4]
fn.geno <- softFilter(fn.geno, filters)
t(summary(softFilterMatrix(fn.geno)))
```

<initial>	readCount	likelihoodRatio	<final>
1021	0	9	0

The default is to evaluate the filters in parallel, but serial evaluation is also supported:

```
fn.geno <- resetFilter(fn.geno)
fn.geno <- softFilter(fn.geno, filters, serial = TRUE)
t(summary(softFilterMatrix(fn.geno)))
```

<initial>	readCount	likelihoodRatio	<final>
1021	0	0	0

dbSNP concordance

Import a VRanges from (p53) dbSNP VCF:

```
vcfPath <- system.file("extdata", "dbsnp-p53.vcf.gz",
                       package = "VariantToolsData")
param <- ScanVcfParam(fixed = "ALT", info = NA, geno = NA)
dbSNP <- as(readVcf(vcfPath, param, genome = "TP53_demo"),
            "VRanges")
dbSNP <- dbSNP[!isIndel(dbSNP)]
```

And annotate the stacked variants for concordance:

```
stacked.variants$dbSNP <- stacked.variants %in% dbSNP
xtabs(~ dbSNP + expected.freq, mcols(stacked.variants))
```

	0	0.25	0.5	0.75	1
FALSE	2233	25	0	0	0
TRUE	917	3497	2023	891	924

Replication over the samples

Tabulate the stacked variants over the samples:

```
tabulated.variants <- tabulate(stacked.variants)
xtabs(~ dbSNP + sample.count, mcols(tabulated.variants))
```

	1	2	3
FALSE	1473	241	101
TRUE	116	435	2422

Visualizing putative FPs: IGV

IGV is an effective tool for exploring alignment issues and other variant calling anomalies; SRAdb drives IGV from R.
To begin, we create a connection: #+begin$_{src}$ R library(SRAdb) startIGV("lm") sock <- IGVsocket() #+end$_{src}$ R

Exporting our calls as VCF

IGV will display variant calls as VCF:

```
mcols(merged.variants) <- NULL
vcf <- writeVcf(sort(merged.variants),
                "merged.variants.vcf",
                index = TRUE)
vcf <- tools::file_path_as_absolute(vcf)
```

Creating an IGV session

Create an IGV session with our VCF, BAMs and custom p53 genome:

```
extdata <- system.file("extdata",
                       package = "VariantToolsTutorial")
bams <- tools::list_files_with_exts(extdata, "bam")
p53fasta <- tempfile("p53", fileext = ".fasta")
rtracklayer::export(TP53Genome(), p53fasta)
session <- IGVsession(c(bams, vcf), "session.xml",
                      p53fasta)
```

Load the session:

```
IGVload(sock, session)
```

Browsing regions of interest

IGV will (manually) load BED files as a list of bookmarks:

```
rtracklayer::export(merged.variants, "roi.bed")
```