Variant Calling Algorithms –
in pooled and non-pooled data sets

André Altmann
Department of Neurology
Functional Imaging in Neurodegenerative Disorders Lab
altmann@stanford.edu

Bioinformatics Short Course
08/02/12, Iowa City
Outline

SNP calling in individual DNA samples
- Local Realignment
- Theoretical Background of SNP calling
- Quality Score Recalibration
- Probabilistic SNP calling

SNP calling in pooled DNA samples
- Differences to standard SNP calling
- Algorithms
- Case Study

Summary
SNP Calling Pipeline
Realignment Around Indels

- Indels may lead to artificial SNP calls in the downstream process

  CTA-AGTACGTGCT
  GCCTAA-GTACGTG
  GTCAGG CCT-AAG
  ACTGCCTAGGTCA GG CCTAAAGTGACGTGCTA (reference)

- Typically done: Local Realignment around Indels

- Tools: GATK, SMRA
GATK does the realignment in two steps

1. Identify regions for local realignment

   $> \text{java } -\text{jar GenomeAnalysisTK.jar } \backslash$
   $-T \text{RealignerTargetCreator } \backslash$
   $-R \text{myreference.fasta } \backslash$
   $-I \text{myalignment.bam } -o \text{myrealign.intervals}$

2. Perform local realignment

   $> \text{java } -\text{jar GenomeAnalysisTK.jar } \backslash$
   $-T \text{IndelRealigner } \backslash$
   $-R \text{myreference.fasta } \backslash$
   $-I \text{myalignment.bam } \backslash$
   $-\text{targetIntervals myrealign.intervals } \backslash$
   $-o \text{myrealignment.bam}$
Outline

SNP calling in individual DNA samples
- Local Realignment
- Theoretical Background of SNP calling
- Quality Score Recalibration
- Probabilistic SNP calling

SNP calling in pooled DNA samples
- Differences to standard SNP calling
- Algorithms
- Case Study

Summary
1st SNP Calling Approach

The simplest approach to SNP calling:
- Count alternative high-quality bases at one position

A smarter way is to use the information on the base quality during the calling process

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
<td>A</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>22</td>
<td>C</td>
<td>22</td>
</tr>
<tr>
<td>C</td>
<td>17</td>
<td>C</td>
<td>17</td>
</tr>
<tr>
<td>A</td>
<td>23</td>
<td>A</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>19</td>
<td>C</td>
<td>19</td>
</tr>
</tbody>
</table>
Basic principle of SNP calling using Phred Scores

\[ p(G|E) = \frac{p(E|G)}{p(E)} \cdot p(G) \]

Calling a genotype is done by computing

\[ \hat{G} = \arg\max_G \{ p(E|G) \cdot p(G) \} \]

Especially,

\[ p(E|G) = \prod_i p(e_i|G) \]

\[ p(e_i|G) \text{ are simply rescaled error probabilities} \]
Probabilistic SNP Calling Approach

Basic principle of SNP calling using Phred Scores

\[ p(G|E) = \frac{p(E|G)}{p(E)} \cdot p(G) \]

The genotype prior plays an important role

- Uniform Prior – all genotypes equally likely
- Prior based on known SNPs (e.g., dbSNP, 1000 genomes)
- Prior based on multiple individuals
- Prior based on phasing information
Probabilistic SNP Calling Approach

- Probabilistic SNP calling makes extensive use of quality scores

- We need a good estimation of the base call quality

- As it turns out, the manufacturer’s base calling methods overestimate the quality of their base calls

- Alternative base calling algorithms are available
  - But it is cumbersome to replace the manufacturer’s pipeline

- A way out?
SNP Calling Pipeline

Base Calling

Quality Control

Alignment/Mapping

Alignment Post-Processing

Quality Score Recalibration

Variant and Genotype Calling

Filtering SNP Candidates

Making Sense of SNP Data

Common for all HTS Pipelines

Specific for SNP Calling
One can estimate the base call quality from the data

Idea in a nutshell

- Use empirical mismatch rate between reference and aligned bases as surrogate for quality score
- Algorithms consider
  - Sequenced base
  - Manufacturer’s base quality estimate
  - Position in the sequence read
  - Sequence context
  - Sites with reported SNPs (e.g., dbSNP) are excluded

ATTAAGTACGTGCT
GGCTCAAGTACGTG
GTCGGGCCTTAAG
ACTGCCTAGGTCAGGCCTTAAGTACGTGCTA (reference)
GATK does the recalibration in two steps

1. **Compute Recalibration Scores**

   ```
   $> java –jar GenomeAnalysisTK.jar \
   -T CountCovariates \ 
   -R myreference.fasta \ 
   -I myalignment.bam –knownSites snplist \ 
   –recalFile recalibration.table –standard
   ```
Quality Score Recalibration

# Counted Sites 19451059
# Counted Bases 56582018
# Skipped Sites 82666
# Fraction Skipped 1 / 235 bp

ReadGroup,QualityScore,Cycle,Dinuc,nObservations,nMismatches,Qempirical
SRR006446,11,65,CA, 9, 1,10
SRR006446,11,48,TA, 10, 0,40
SRR006446,11,67,AA, 27, 0,40
SRR006446,11,61,GA, 11, 1,10
SRR006446,12,34,CA, 47, 1,17
SRR006446,12,30,GA, 52, 1,17
SRR006446,12,36,AA,352, 1,25
SRR006446,12,17,TA,182,11,12
SRR006446,11,48,TG,  2, 0,40
SRR006446,11,67,AG,  1, 0,40
SRR006446,12,34,CG,  9, 0,40
SRR006446,12,30,GG, 43, 0,40
ERR001876, 4,31,AG,  1, 0,40
ERR001876, 4,31,AT,  2, 2, 1
ERR001876, 4,31,CA,  1, 0,40
GATK does the recalibration in two steps

1. Compute Recalibration Scores
   $> \text{java} \ –\text{jar} \ \text{GenomeAnalysisTK.jar} \ \\$
   \quad \text{-T} \ \text{CountCovariates} \ \$
   \quad \text{-R} \ \text{myreference.fasta} \ \$
   \quad \text{-I} \ \text{myalignment.bam} \ \text{–knownSites snplist} \$
   \quad \text{–recalFile recalibration.table} \ \text{–standard}$

2. Perform local recalibration
   $> \text{java} \ –\text{jar} \ \text{GenomeAnalysisTK.jar} \ \$
   \quad \text{-T} \ \text{TableRecalibration}$
   \quad \text{-R} \ \text{myreference.fasta} \ $
   \quad \text{-I} \ \text{myalignment.bam} \ $
   \quad \text{–recalFile recalibration.table} \ $
   \quad \text{-o} \ \text{myrecalibrated.bam}$

Quality Score Recalibration
Effect Of Quality Score Recalibration

![Graph showing the effect of quality score recalibration](image)
SNP Calling Pipeline

Base Calling

Quality Control

Alignment/ Mapping

Alignment Post-Processing

Quality Score Recalibration

Variant and Genotype Calling

Filtering SNP Candidates

Making Sense of SNP Data

Common for all HTS Pipelines

Specific for SNP Calling

Input: Images from Sequencer
Tools: Internal HTS System Software
Output: Base- or Color-Sequence and Quality Scores -> e.g. fastq

Input: Read Data -> e.g. fastq
Tools: SolexaQA, FastQC, PRINSEQ
Output: Quality Report and Filtered Reads -> e.g. fastq

Input: Filtered Read Data
Tools: BWA, MAQ, Stampy,Bowtie, SHRIMP2, bfast
Output: SAM, BAM and Mapping Statistics

Input: SAM/BAM
Tools: samtools, Picard, SMRA, GATK
Output: SAM/BAM

Input: SAM/BAM
Tools: SOAPsnp, GATK
Output: SAM/BAM

Input: SAM/BAM
Tools: SOAPsnp, MAQ, samtools, GATK, Beagle
Output: vcf

Input: vcf
Tools: GATK, samtools, VCF tool
Output: vcf
The Actual SNP calling

- After having realigned and recalibrated bam files
- There are many tools
- Tools are under constant development/improvement
- Most popular are probably GATK and samtools
- Let’s first have a look at the output files
Variant Call Format (VCF)

http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41

BCF is the binary VCF format

Manipulate VCF files with VCFtools (http://vcftools.sourceforge.net/)
Call SNPs from a single alignment

```bash
$> java –jar GenomeAnalysisTK.jar \\
-T UnifiedGenotyper \ 
-R myreference.fasta \ 
-I myalignment.bam \ 
--dbSNP knownsnps.vcf \ 
-o mysnps.vcf
```
Better results with prior derived from multiple individuals

```shell
$> java –jar GenomeAnalysisTK.jar \\
    -T UnifiedGenotyper \\
    -R myreference.fasta \\
    -I myalignment.bam –I myalignment2.bam \ 
    –I … \ 
    --dbSNP knownsnps,vcf \ 
    –o mysnps.vcf
```
Best results when exploiting LD information

Recommendation: “Beagle” an impute tool

GATK can generate input files for Beagle

```bash
$> java –jar GenomeAnalysisTK.jar \
-T ProduceBeagleInput \
-R myreference.fasta \
-V mycaledSNPs.vcf \
-o beagleinput.txt
```

Run Beagle on beagleinput.txt
Call SNPs from a single alignment:

```
$> \text{samtools mpileup} \ -uf \text{myreference.fasta \ myalignment.bam} \ > \ \text{tmp\_output.bcf}
```

```
$> \text{bcftools view} \ -bv\text{cg} \ \text{tmp\_output.bcf} \ > \ \text{raw.bcf}
```

```
$> \text{bcftools view} \ > \ \text{raw.vcf}
```

```
$> \text{vcfutils.pl varFilter} \ -D100 \ \text{raw.vcf} \ > \ \text{my\_samtool\_snps.vcf}
```
The Actual SNP calling – samtools

```bash
$> samtools mpileup -uf myreference.fasta \ myalignment.bam | bcftools view -bvcg > raw.bcf

$> bcftools view > raw.vcf

$> vcfutils.pl varFilter -D100 raw.vcf > my_samtool_snps.vcf
```
The Actual SNP calling – samtools

```bash
$> samtools mpileup -uf myreference.fasta \ myalignment.bam | bcftools view -bvcg > raw.bcf

$> bcftools view | vcfutils.pl varFilter \ -D100 > my_samtool_snps.vcf

Using prior from multiple individuals:
$> samtools mpileup -uf myreference.fasta \ myalignment.bam myalignment2.bam ... | bcftools view -bvcg > raw.bcf
```
Add newly identified SNPs to the list of known SNPs and redo recalibration.
InDel Detection

- **InDel Calling**

- **Two Types of InDels**
  - Large InDels (i.e., structural variants)
  - Short InDels (a few bp long)

- **Short InDels show up within aligned reads**
  - GATK (InDel calling reached stable status recently)
  - samtools
  - Dindel (http://www.sanger.ac.uk/resources/software/dindel/)

- **Large InDels are detectable**
  - Via analysis of insert size length
  - Software Pindel (https://trac.nbic.nl/pindel/)
  - Via analysis of depth of coverage (Copy Number Variants)
**InDel Detection**

- **Short InDels**
  - CT-AAGTACGTGCT
  - GCCT-AAGTACGTG
  - GTCAGGCCT-AAG
  - ACTGCCTAGGTACGGCCTTAAGTACGTGCTA (reference)

- **Large InDels (Structural Variants)**

  Xi et al. (2011)
  doi:10.1093/bfgp/elq025
A Word On Working With GATK

GATK requires your data to comply to certain standards

- The reference must be karyotypically ordered
- Read groups must be stored within sam/bam

Best way

- Download reference sequence from GATK

If data is already aligned or RG is missing

- Use Picard!
- Reorder alignment
- Add/remove RGs
Outline

SNP calling in individual DNA samples
- Local Realignment
- Theoretical Background of SNP calling
- Quality Score Recalibration
- Probabilistic SNP calling

SNP calling in pooled DNA samples
- Differences to standard SNP calling
- Algorithms
- Case Study

Summary
SNP calling in pooled datasets

Finding rare SNPs
- Common Disease – Rare Variant - Hypothesis

Whole genome / exome re-sequencing for 1,000s of individuals unrealistic (within one lab)

Ways for reducing the sequencing costs
- Focus on smaller region to sequence
- Whole Exome instead of Whole Genome
- Target gene(s) instead of whole exome
Focusing on a target region drastically reduced the sequencing costs, but…

… each sample has to be prepared for sequencing: Library preparation
Focusing on a target region drastically reduced the sequencing costs, but…

… barcoding doesn’t help too much here:
From common to rare variants

Focusing on a target region drastically reduced the sequencing costs, but…

… DNA pools help reducing the cost (and labor):
The reduction in costs and labor comes at a price …

Loss of information
- Which subject has which mutations?
- Some information can be retained due to clever experimental design – Tool DNA Sudoku

Increase of noise
- No PCR duplicate removal
- Rare variants (1 heterozygous) might be indistinguishable from sequencing errors
- Even worse: sequence error rate is not uniform
Pooled Sequencing - Coverage is uneven – but conserved
SNP Calling Pipeline – Changes in Pooled Samples

No PCR
duplicate removal

No re-alignment around indels

No quality score recalibration
Finding (rare) variants in DNA pools

Frequency of the most frequent alternative allele

Using statistics is better than a hard cutoff
Finding (rare) variants in DNA pools

- **Model sequencing errors as a Poisson distribution**
  \[
  \text{pois}(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}
  \]
  - Expected sequencing errors
  - Observed allele count

- **Calculate the likelihood that the observed allele counts were generated by sequencing errors**
  \[
  P_i^{\text{pois}} = 1 - \sum_{k=0}^{r_i-1} \text{pois}(k; \lambda_i)
  \]

- **Problem: the sequencing error rate is not uniform**
The sequencing error rate is quite conserved between different pools.
Finding (rare) variants in DNA pools

The sequencing error rate is quite conserved between different pools.

So we can analyze the difference in allele frequencies.

Altmann, André – SNP calling in un-pooled and pooled MPS data – 08/02/2012
Finding (rare) variants in DNA pools

The Skellam distribution models the difference of two Poisson distributions:

\[ \text{skel}(k; \mu_1; \mu_2) = e^{-(\mu_1+\mu_2)} \left( \frac{\mu_1}{\mu_2} \right)^{k/2} I_{|k|}(2 \sqrt{\mu_1 \mu_2}) \]

Calculate the likelihood that the observed difference in allele counts was generated by chance:

\[ P_{i,a,b}^{\text{skel}} = 1 - \sum_{k=-\infty}^{d_i-1} \text{skel}(k; \mu_i^a; \mu_i^b) \]

If the resulting p-value is sufficiently small, the position harbors a variant.

Expected sequencing errors Poisson 1
Observed difference in allele count
Expected sequencing errors Poisson 2
Finding (rare) variants in DNA pools

- Power calculation: Poisson vs Skellam

- Error rate is kept constant at $2.7 \times 10^{-3}$ and the allele frequencies vary from 1 to 4 heterozygous alleles in 150 individuals
Finding (rare) variants in DNA pools

- **Power calculation**: Poisson vs Skellam

- **Allele frequency** is kept fixed at 0.33% and the error rate varies from 0.1% to 0.5%
Finding (rare) variants in DNA pools

Using power calculations it can be shown that
- The Skellam distribution is more suited to find rare SNPs than the Poission distribution
- The Skellam distribution is less prone to the sequencing error rate of the sequencer than the Poission distribution

This idea is implemented in vipR
- Focus only on the most frequent alternative allele
- SNP has to be found in reads aligned in forward and in reverse direction
- One library compared to all remaining ones

vipR freely available via sourceforge:
http://sourceforge.net/projects/htsvipr/
Running vipR

vipR comes in two parts

1. Java/Python tool for pileup -> vipr.input
2. vipR R program for snp calling

Before running vipR: convert alignment to pileup format

Samtools version 0.1.7a
$> samtools pileup –s –f ref.fasta aln.bam

Current Samtools version
$> samtools mpileup –s –d 1000000 –f ref.fasta \ aln.bam
Running vipR

- vipR comes in two parts

1. Java/Python tool for pileup -> vipr.input
2. vipR R program for snp calling

- Pileup to vipr.input:

  $> \text{java pileup2vipr --c minCover pool1.pile pool1.txt}$

- Launch R, and run vipR:

  R>source(“vipR.R”)
  R>my.pools <- vipr.loadData(c(“pool1.txt”, “pool2.txt”))
  R> vipr.run(my.pools, nhap, “mysnps.vcf”)
Finding (rare) variants in DNA pools – other tools

- **VarScan**
  - varscan.sourceforge.net

- **CRISP**
  - https://sites.google.com/site/vibansal/software/crisp
  - Computes p-values based on $2 \times k$ contingency tables
  - Uses quality scores

- **SNPseeker/SPLINTER**
  - Illumina platform only
  - Uses spiked in synthetic library to estimate precise error model for data at hand
Outline

- SNP calling in individual DNA samples
  - Local Realignment
  - Theoretical Background of SNP calling
  - Quality Score Recalibration
  - Probabilistic SNP calling

- SNP calling in pooled DNA samples
  - Differences to standard SNP calling
  - Algorithms
  - Case Study

Summary
Case study: TMEM132D

Panic disorder:

- Sudden episodes of acute anxiety and fear occurring without a stimulus

The gene locus of TMEM132D is associated with anxiety phenotypes

- Identified in a GWAS (Erhardt et al., Mol Psychiatry, 2010)
- Covers 834 kb of sequence on chromosome 12
- Comprises 8 exons
Case study: TMEM132D

Parts of TMEM 132D were sequenced in 300 healthy controls and 300 affected individuals.

Individuals were distributed over four equally sized DNA pools.

The regions were enriched using custom primers and long-range PCR and sequenced using a SOLiD 3+.
Case study: TMEM132D

Sample processing:

For each library
1. Quality control
2. Read trimming
3. Alignment to chromosome 12
4. SNP finding with vipR

vipR identified 371 SNPs – 247 not in dbSNP132
82 potential SNPs were validated with an independent technique: MALDI-TOF

The performance of vipR was compared to three other methods:
- Poisson, VarScan (version 1), and CRISP (2010 version)
Case study: TMEM132D

These data allow us to compute objective performance measures
- E.g. sensitivity, specificity, …

<table>
<thead>
<tr>
<th>Position</th>
<th>Cases1 vipR</th>
<th>Cases1 MALDI</th>
<th>Controls1 vipR</th>
<th>Controls1 MALDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td><strong>SNP</strong></td>
<td><strong>SNP</strong></td>
<td>-</td>
<td>SNP</td>
</tr>
<tr>
<td>1266</td>
<td><strong>SNP</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1488</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

True positive
False negative
False positive
True negative
Case study: TMEM132D

<table>
<thead>
<tr>
<th></th>
<th>vipR</th>
<th>CRISP</th>
<th>Poisson</th>
<th>VarScan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensitivity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cases 1</td>
<td>0.77</td>
<td>0.77</td>
<td>0.72</td>
<td>0.81</td>
</tr>
<tr>
<td>Cases 2</td>
<td>0.76</td>
<td>0.76</td>
<td>0.52</td>
<td>0.72</td>
</tr>
<tr>
<td>Controls 1</td>
<td>0.81</td>
<td>0.81</td>
<td>0.56</td>
<td>0.85</td>
</tr>
<tr>
<td>Controls 2</td>
<td>0.88</td>
<td>0.88</td>
<td>0.72</td>
<td>0.92</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>0.80</td>
<td>0.80</td>
<td>0.60</td>
<td>0.82</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>vipR</th>
<th>CRISP</th>
<th>Poisson</th>
<th>VarScan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specificity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cases 1</td>
<td>0.89</td>
<td>0.79</td>
<td>0.89</td>
<td>0.59</td>
</tr>
<tr>
<td>Cases 2</td>
<td>0.91</td>
<td>0.75</td>
<td>0.79</td>
<td>0.53</td>
</tr>
<tr>
<td>Controls 1</td>
<td>0.95</td>
<td>0.75</td>
<td>0.89</td>
<td>0.55</td>
</tr>
<tr>
<td>Controls 2</td>
<td>0.95</td>
<td>0.72</td>
<td>0.88</td>
<td>0.54</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>0.92</td>
<td>0.75</td>
<td>0.86</td>
<td>0.55</td>
</tr>
</tbody>
</table>

- vipR is as sensitive as the other methods
  - i.e. We find as many real SNPs
- ... but at a pronounced gain in specificity
  - i.e. while producing fewer “false alarms”
By varying one of the parameters of the algorithms on can obtain a ROC curve

Parameter to vary: # of haplotypes in a DNA pool

ROC curve for the pool-wise performance
Case study: TMEM132D

- Runtime required for analyzing one to four libraries

- Runtime (in sec) on a single Intel core @ 2.67GHz and 6GB memory
Finding (rare) variants in DNA pools – other tools

- **VarScan**
  - varsan.sourceforge.net
  - Current version 2.

- **CRISP**
  - https://sites.google.com/site/vibansal/software/crisp
  - last update July 2012
  - Has been improved in runtime and accuracy

- **Due to updates advantage of vipR might be less pronounced**
InDel Detection In Pooled Samples

- **vipR** can detect short InDels
  - High false-positive rate
  - (Probably) high false-negative rate

- **Alternative tools:**
  - SPLINTER, requires spiked in synthetic library
Further Reading

SeqAnswers Forum
- http://seqanswers.com/
- Wiki with software tools

GATK best practice guidelines

Literature:
Nielsen et al. (2011) *Nat Rev Gen*  doi:10.1038/nrg2986
Altmann et al. (in press) *Human Genetics*
End
Questions?