Esercitazione di genotipizzazione

Corso di Genomica Computazionale, Laurea Specialistica in Genomica Funzionale, Universita' di Trieste, Fabrizio Mafessoni, Tutor: Daniela Eugenia Nerelli

Installiamo bcftools

Usando conda

```
conda install -c bioconda bcftools
```

Oppure con brew (per mac) o apt per linux/windows

```
sudo apt install bcftools
```

```
brew install bcftools
```

Genotipizzazione a singolo campione

Scarichiamo le prime reads allineate (bam) di un individuo caucasico dal 1000 Genomes project. Utilizziamo solo il cromosoma 20 per velocizzare i calcoli.

```
wget
https://1000genomes.s3.amazonaws.com/phase3/data/NA12878/align
ment/NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.ba
m
```

Indicizziamo il bam

```
samtools index
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam
```

Scarichiamo il genoma di referenza

```
wget https://storage.googleapis.com/gatk-best-
practices/somatic-b37/Homo_sapiens_assembly19.fasta
```

Per lavorare piu velocemente concentriamoci solo sul cromosoma 20, indicizzando il genoma di referenza ed estraendo il cromosoma 20

```
samtools faidx Homo_sapiens_assembly19.fasta
```

```
samtools faidx Homo_sapiens_assembly19.fasta 20 > chr20.fa
```

```
samtools faidx chr20.fa
```

Esploriamo il nostro file bam e verifichiamo che abbiamo solo reads sul cromosoma 20 con idxstats.

```
samtools view
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam |
head
```

```
samtools idxstats
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam
```

Osserviamo le nostre reads nel bam in formato mpileup

```
samtools mpileup
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam
```

Notate che mpileup **non chiama genotipi**.
Costruisce una **rappresentazione base-per-base** di cosa mostrano le letture rispetto al riferimento in un dato sito. È quindi un formato di **evidenza grezza**, che poi viene passato a **bcftools call** per fare la vera chiamata di variante/genotipo.

Significato delle colonne:

Colonna	Nome	Significato
1	Chromosome	Nome del contig (es. 20, chr20)
2	Position	Posizione 1-based sul riferimento
3	Reference base (REF)	La base attesa dal genoma di riferimento
4	Depth (DP)	Numero totale di letture che coprono la posizione
5	READS column	Cosa mostrano le letture
6	QUAL column	Qualità dei nucleotidi (ASCII 33-based, come FASTQ)

Notate che in READ:

A C G T (maiuscole) base letta **su filamento forward**

a c g t (minuscole) base letta **su filamento reverse**

$ indica la fine di una lettura di una sequenza, mentre ^ l'inizio.

Riguardo agli INDELS invece:

Esempio Significato

+3ACT inserzione di lunghezza 3 con sequenza ACT

-2GA delezione di lunghezza 2 di GA

Usiamo il Pileup di bcftools per creare le genotype likelihood, senza ancora chiamare il campo GT (genotipo)

```
bcftools mpileup -f chr20.fa
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam |
grep -v '#' | head
```

Queste hanno il seguente formato:

Campo	Nome	Significato
PL	Phred-scaled Genotype Likelihoods	3 numeri (0/0, 0/1, 1/1)
GL (a volte)	Genotype Likelihoods log10	più rari; PL è lo standard
DP	Depth	numero letture
AD	Allele Depths	numero REF, ALT

Ad esempio:

GT:PL:DP:AD

0/1:135,0,161:11:6,5

Valore	**Significato**
GT = 0/1	Chiamato come eterozigote
PL = 135,0,161	Likelihoods **PHRED** per 0/0, 0/1, 1/1
DP = 11	11 letture totali
AD = 6,5	6 REF e 5 ALT

Come si leggono i PL (phred-scaled likelihoods)

PL=−10 $\log_{10}$P(dati|genotipo)PL

→ più piccolo = più probabile.

Nell'esempio:

- 0/0 ha PL = 135 → molto improbabile
- **0/1 ha PL = 0 → genotipo più probabile**
- 1/1 ha PL = 161 → molto improbabile

Mentre mpileup calcola quanto sono probabili i dati per ogni genotipo (0/0,0/1,1/1),
call sceglie il genotipo più probabile utilizzando le genotype likelihood calcolate da mpileup.
Dunque, chiamiamo le varianti, aggiungendo il genotipo GT:

```
bcftools mpileup -Ou -f chr20.fa -a FORMAT/DP,FORMAT/AD \
  NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam \
| bcftools call -mv -Oz -o calls.vcf.gz
```

```
bcftools index calls.vcf.gz
```

Creiamo una piccola tabella dei genotipi

```
bcftools query -f '%CHROM\t%POS\t%REF\t%ALT\t%QUAL\t[%DP]\n'
calls.vcf.gz | head
```

Usiamo awk per contare SNPs e INDELs

```
bcftools view -H calls.vcf.gz \

| awk '{if(length($4)==1 && length($5)==1) snp++; else ind++}
END{print "SNP:",snp,"INDEL:",ind}'
```

Usiamo bcftools per filtrare il nostro vcf

```
bcftools view -g het calls.vcf.gz | grep -v '#' | head

bcftools view -i 'FORMAT/DP>=10 && QUAL>=30' calls.vcf.gz |
grep -v '#' | head
```

Genotipizzazione multi-campione

Per ora abbiamo genotipizzato un singolo campione. Proviamo ora a fare una genotipizzazione multi-campione scaricando altri individui dai 1000 genomes:

```
wget
https://1000genomes.s3.amazonaws.com/phase3/data/NA12878/align
ment/NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.ba
m

wget
https://1000genomes.s3.amazonaws.com/phase3/data/HG00096/align
ment/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.ba
m

wget
https://1000genomes.s3.amazonaws.com/phase3/data/HG00110/align
ment/HG00110.chrom20.ILLUMINA.bwa.GBR.low_coverage.20130415.ba
m

samtools index
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam

samtools index
HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam

samtools index
HG00110.chrom20.ILLUMINA.bwa.GBR.low_coverage.20130415.bam
```

Esercizio supplementare (provate a implementare questi ultimi comandi in modo migliore con un for loop).

Creiamo una piccola lista dei nostri bam

```
printf "%s\n" \
  NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam \
  HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam \
  HG00110.chrom20.ILLUMINA.bwa.GBR.low_coverage.20130415.bam \
  > bamlist.txt
```

Genotipizziamo i nostri tre individui. Notate la sintassi. Esplorate la funzione delle varie flag con `bcftools --help`

```
bcftools mpileup -Ou -f chr20.fa \
  -a FORMAT/DP,FORMAT/AD,FORMAT/ADF,FORMAT/ADR \
  -b bamlist.txt \
| bcftools call -m -v -Oz -o trio.recall.vcf.gz
bcftools index trio.recall.vcf.gz
```

e rigenotipizziamo il nostro individuo singolo

```
bcftools mpileup -Ou -f chr20.fa \
  -a FORMAT/DP,FORMAT/AD,FORMAT/ADF,FORMAT/ADR \
  NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam \
| bcftools call -m -v -Oz -o NA12878.single.recall.vcf.gz
bcftools index NA12878.single.recall.vcf.gz
```

Usiamo il comando isec per ottenere una comparazione delle nostre chiamate a singolo e multi-campione:

```
bcftools isec -p isec_out NA12878.single.recall.vcf.gz
trio.recall.vcf.gz
```

Notate che gli output (nella cartella isec_out) hanno questa struttura:

0000.vcf → varianti solo in file #1

0001.vcf → variant solo in file #2

0002.vcf → varianti in entrambi (record come in file #1)

0003.vcf → varianti in entrambi (record come in file #2)

Esercizi:

1) Esplorate le differenze tra le chiamate per i differenti individui usando awk e i comandi bash di vostra conoscenza. Quante varianti sono state perse, e quante guadagnate con la genotipizzazione multi-campione? Suggerimento: testate tutti i campioni usando un `for` loop.
2) Comparate le posizioni da voi trovate con i file samtools mpileup. Che tipo di varianti sono scomparse nella genotipizzazione multi-campione? Quali sono state guadagnate?
3) Scaricate l'intero dataset del 1000 genomes per il cromosoma 20 ed esploratelo con bcftools. Ricordate di indicizzare. Estraete i genomi dei campioni da voi analizzati individualmente e comparatene i genotipi. Cosa notate? Avete perso o guadagnato varianti? Per scaricare i dati usate: `wget` https://hgdownload.cse.ucsc.edu/gbdb/hg19/1000Genomes/phase3/ALL.chr20.phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.vcf.gz
   E per estrarre i vostri campioni usando la flag -S di bcftools
   ```
   printf "NA12878\nNA12891\nNA12892\n" > samples.txt
   bcftools view -S samples.txt -Oz \

   ALL.chr20.phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.vcf.gz \
     > trio.chr20.1000G.vcf.gz
   ```
   Opzionalmente, per velocizzare, potete estrarre solo varianti in una data regione con
   ```
   REGION=20:10000000-10040000
   bcftools view -r $REGION -Oz -o trio.$REGION.vcf.gz
   trio.chr20.1000G.vcf.gz
   bcftools index trio.$REGION.vcf.gz
   ```
4) Chiamate i genotipi per il cromosoma 20 con bcftools per i genomi presenti in: http://ftp.eva.mpg.de/denisova/BAM/human/

Soluzioni

1) Prima di tutto generiamo le chiamate per tutti gli individui

```
for i in NA12878 HG00096 HG00110;do
echo $i
MYBAM=${i}.chrom20.ILLUMINA.bwa.*.low_coverage.*.bam
bcftools mpileup -Ou -f chr20.fa \
  -a FORMAT/DP,FORMAT/AD,FORMAT/ADF,FORMAT/ADR \
  ${MYBAM} | bcftools call -m -v -Oz -o ${i}.single.recall.vcf.gz
bcftools index ${i}.single.recall.vcf.gz
done
```

Ora compariamo tutti questi vcf individualmente alle chiamate dei genotipi fatti su tutti e tre gli individui simultaneamente. Prima di tutto notiamo che per tutti, sono molte piu' le varianti chiamate per i genotipi chiamati simultaneamente.

```
for i in NA12878 HG00096 HG00110;do
echo ${i}
mkdir -p isec_out/${i}
bcftools isec -p isec_out/${i} ${i}.single.recall.vcf.gz trio.recall.vcf.gz
VAR_SINGLE=$( cat isec_out/${i}/0000.vcf | grep -v '#' | wc -l)
VAR_TRIO=$( cat isec_out/${i}/0001.vcf | grep -v '#' | wc -l)
echo "Varianti presenti solo in ${i} ${VAR_SINGLE}"
echo "Varianti presenti solo nel file trio ${VAR_TRIO}"
done
```

Tuttavia questo potrebbe essere dovuto sia al fatto che nel file trio ci sono piu' individui, sia ad un maggiore potere nel chiamare le varianti simultaneamente. Noi siamo interessati solamente alla seconda componente. Pertanto, escludiamo ora dal vcf trio le chiamate di varianti che sono presenti solo negli altri individui e non nell'individuo target, al fine di esaminare appropriatamente solo l'effetto della chiamata a singolo individuo verso quella multi-individuo. Pertanto, cambiamo il modo di calcolare VAR_TRIO, filtrando le varianti desiderate (0/1, 1/1, e altre, ma non 0/0 o varianti non chiamate):

```
VAR_TRIO_FILTERED=$( bcftools view -s ${i} isec_out/${i}/0001.vcf | awk
'{if (substr($10,1,3)!="0/0" && substr($10,1,3)!="./."){print}}' | wc -l
)
```

Noterete che il numero di varianti chiamate è molto piu' alto quando i genotipi sono chiamati simultaneamente per tutti gli individui, anche guardando alle varianti per cui troviamo supporto negli individui singoli. Queste sono tutte varianti che non avremmo potuto chiamare con confidenza se non avessimo effettuato una chiamata multi-sample.

```
for i in NA12878 HG00096 HG00110;do
echo ${i}
VAR_SINGLE=$( cat isec_out/${i}/0000.vcf | grep -v '#' | wc -l)
VAR_TRIO=$( cat isec_out/${i}/0001.vcf | grep -v '#' | wc -l)
VAR_TRIO_FILTERED=$( bcftools view -s ${i} isec_out/${i}/0001.vcf | awk
'{if (substr($10,1,3)!="0/0" && substr($10,1,3)!="./."){print}}' | wc -l
)
echo "Varianti presenti solo in ${i} ${VAR_SINGLE}"
echo "Varianti presenti solo nel file trio ${VAR_TRIO}"
echo "Varianti presenti in $i e chiamate nel file trio
${VAR_TRIO_FILTERED}"
done
```

2) Ora esploriamo piu' in dettaglio come vengono ottenute queste genotype likelihood. Per questo, esaminiamo i file samtools mpileup e compariamoli con i bcftools mpileup. Esaminiamo le prime 2 posizioni

```
samtools mpileup
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam |
head -2
```

```
bcftools mpileup -f chr20.fa
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam |
grep -v '#' | head -2
```

Notiamo rispettivamente 4 G per la prima e 4 T per la seconda posizione (notate come nel samtools mpileup le prime posizioni di ogni read siano sempre indicate dal simbolo ^>, e come osserviamo read depth DP=4 nel bctools mpileup). Non sorprendentemente la Phred-scaled genotype Likelihood piu' alta è quella degli omozigoti (G/G) con PL=0 (quasi totale certezza per G/G). Per la prima posizione (chr20:59993) dovreste notare dei campi PL circa:
88,12,0,88,12,88
i quali si riferiscono alle PL di genotipi 0/0,0/1,1/1,0/2,1/2,2/2 con allele referenza N e alternativo G (notate che la N risulta referenza perché non avevamo definito una referenza di input nel comando mpileup, quindi G/G corrisponde ad 1/1).
Questo non è una caso molto interessante da esplorare. Proviamo invece ad esplorare qualche genotipo inferito come eterozigote. Iniziamo da qualcuno inferito come eterozigote sia nella genotipizzazione single-sample che multi-sample, e prendiamo il primo:

```
cat isec_out/NA12878/0003.vcf | grep '0/1' | head -1
```

```
#20      61795   .       G       T       265.217 .       DP=30;VDB=0.522432;SGB=-
1.94008;RPBZ=-1.10228;MQBZ=0.725476;MQSBZ=-0.901388;BQBZ=-
0.484284;SCBZ=0;MQ0F=0;AC=3;AN=6;DP4=9,10,4,6;MQ=58 GT:PL:DP:ADF:ADR:AD
0/1:135,0,161:11:5,2:1,3:6,5    1/1:171,15,0:5:0,2:0,3:0,5      0/0:0,39,255:13:4,0:9,0:13,0
```

Notiamo come il primo campione abbia genotipo eterozigote 0/1, il secondo 1/1 ed il terzo 0/0. Questa variante appare quindi nel 50% degli alleli.
Ma come appare nel samtools mpileup?

```
for i in NA12878 HG00096 HG00110;do
echo ${i}
samtools mpileup ${i}.chrom20.ILLUMINA.bwa.*.low_coverage.*.bam | awk -v
SNP=61795 '{if ($2==SNP){print;exit}}' | tr 'acgt' 'ACGT'
done
Noterete che nei tre individui i samtools mpileup sono
#GTGGGGTTTGT
#TTTTT
#GGGGGGGGGGGGG
```

Nel primo individuo NA12878 ci sono 6 G e 5 T. Guardiamo le genotypes likelihoods:

```
i=NA12878;tabix ${i}.single.recall.vcf.gz
tabix ${i}.single.recall.vcf.gz  20:61795-61795
#GT:PL:DP:ADF:ADR:AD     0/1:135,0,161:11:5,2:1,3:6,5
```

Vediamo che la genotype likelihood in phred scale è 135 per 0/0 (cioè circa 10^-14) e 10^-16 per 1/1, mentre circa 1 per 0/1. Ha senso? Per verificarlo, proviamo a costruire un programma di genotipizzato molto semplice per il singolo campione, usando awk!!
Prima di tutto, chiamiamo samtools mpileup fornendo il genoma di referenza chr20.fa con la flag -f, di modo tale che l'allele di referenza venga mostrato come "." o ",".

Eliminiamo anche caratteri speciali come quelli di inizio reads o indel con `sed (sed -E 's/\^./ /g; s/\$//g; s/[+-][0-9]+[ACGTNacgtn]+//g' )`.

Creiamo poi un piccolo script awk chiamato count_ref_alt.awk (in appendice) che semplicemente conti gli alleli referenza e alternativi, di modo che dia un output che dia cromosoma,posizione, allele REF, conto REF, allele ALT, conto ALT, conto totale (o Read Depth), e poi un PASS quando il sito è biallelico (per semplicità evitiamo di calcolare le genotype likelihood per siti triallelici):

```
samtools mpileup
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam -f
chr20.fa | sed -E 's/\^./ /g; s/\$//g; s/[+-][0-
9]+[ACGTNacgtn]+//g' | awk -f count_ref_alt.awk | head
#20 59993 N 0 G 1 1 PASS
#20 59994 N 0 T 4 4 PASS
```

A questo punto aggiungiamo uno piccolo script awk simple_genotyper.awk (nell'appendice), che calcoli i genotype likelihood assumendo un semplice modello binomiale con errore costante dell'1%. Il cuore del genotyper consiste delle seguenti righe:

```
    pRR = e
    pRA = 0.5
    pAA = 1 - e
    lnL_RR = k*log(pRR) + (n-k)*log(1-pRR)
    lnL_RA = k*log(pRA) + (n-k)*log(1-pRA)
    lnL_AA = k*log(pAA) + (n-k)*log(1-pAA)
```

Le prime 3 definiscono il nostro modello di errore: quella che definivamo come p nelle slides, e che non è altro che la probabilità di campionare un allele alternativo (0.5 per un allele eterozigote, ed invece "1-e" ed "e" per gli omozigoti, con "e" il tasso di errore).

Le seconde tre definiscono la distribuzione binomiale. Notate che non consideriamo qui il coefficiente binomiale (visto che per ogni sito nel genoma cambia, ma è uguale per tutti i genotipi e quindi si può semplificare) e che calcoliamo la log(likelihood) invece della likelihood. Quindi:

```
samtools mpileup
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam -f
chr20.fa | sed -E 's/\^./ /g; s/\$//g; s/[+-][0-
9]+[ACGTNacgtn]+//g' | awk -f count_ref_alt.awk | awk -f
simple_genotyper.awk | grep 61795

#20      61795   G       T       6       2       8       0/1
G/T     GL=16.18,-0.00,96.00
```

Notiamo che stimiamo con confidenza che il genotipo piu' probabile sia G/T come atteso. Notiamo che le probabilità di 0/0 e 1/1 sono piu' alte che in bcftools. Questo è

probabilmente dato che le basi avevano tassi di errore verosimilmente molto piu' basse dell'1%. Vediamo come le genotype likelihood sarebbero con un piu' realistico 0.001:

```
samtools mpileup
NA12878.chrom20.ILLUMINA.bwa.CEU.low_coverage.20121211.bam -f
chr20.fa | sed -E 's/\^./ /g; s/\$//g; s/[+-][0-
9]+[ACGTNacgtn]+//g' | awk -f count_ref_alt.awk | awk -v
e=0.001 -f simple_genotyper.awk | grep 61795
```

```
#20      61795   G       T       6       2       8       0/1
G/T     GL=35.94,-0.00,155.93
```

Il valore per 1/1 è piu' alto, come ci attendiamo con errori piu' bassi che quindi rendono ancora piu' implausibile che il sito appaia eterozigote per errore. Notate che bcftools usa i valori di errore osservati per ogni base, non uno uniforme ed assunto a priori come nel nostro caso, pertanto i valori non possono coincidere.

3) Ora compariamo i nostri genotipi con quelli presenti nel dataset dei 1000 genomi. Estraiamo solo i nostri tre individui e solo le varianti per cui almeno uno di questi sia eterozigote o alternativo.

```
bcftools view -s NA12878,HG00096,HG00110
ALL.chr20.phase3_shapeit2_mvncall_integrated_v5a.20130502.geno
types.vcf.gz | bcftools view -i 'COUNT(GT="het" ||
GT="alt")>0' | bgzip -f > trio1000g.chr20.vcf.gz
```

```
tabix trio1000g.chr20.vcf.gz
```

Ora compariamo nuovamente con `bcftools isec`

```
mkdir -p isec_out/1000g
```

```
bcftools isec -p isec_out/1000g trio1000g.chr20.vcf.gz
trio.recall.vcf.gz
```

Notiamo che ora la maggior parte delle posizioni sono condivise tra il nostro trio e quelle del trio chiamate anche nel 1000 genomes dataset, mentre meno varianti sono uniche. Sono leggermente piu' quelle chiamate nel 1000 genomes. Ora, come effettuato prima, potrebbe essere interessante vedere come appaiono quelle esclusive dei due datasets.

```
wc -l isec_out/1000g/000*.vcf
```

```
#   30332 isec_out/1000g/0000.vcf
#   27672 isec_out/1000g/0001.vcf
#  101940 isec_out/1000g/0002.vcf
#  101801 isec_out/1000g/0003.vcf
```

Appendice 1: file count_ref_alt.awk

```
#!/usr/bin/awk -f
# Input: cleaned, uppercased mpileup lines
# Output:
# chr  pos  REF  nREF  ALT_base  nALT  depth  status
/^#/ { next }   # skip any header lines
{
    chr   = $1
    pos   = $2
    REF   = $3       # already uppercased by tr
    bases = $5       # qualities in $6 are ignored

    # initialize counts
    nREF = nA = nC = nG = nT = 0

    # count REF (., ,) and ALT bases (A/C/G/T)
    for (i = 1; i <= length(bases); i++) {
        c = substr(bases, i, 1)
        if (c == "." || c == ",") {
            nREF++
        } else if (c == "A") {
            nA++
        } else if (c == "C") {
            nC++
        } else if (c == "G") {
            nG++
        } else if (c == "T") {
            nT++
        }
        # we ignore N and anything else
    }

    depth = nREF + nA + nC + nG + nT

    # determine ALT allele if at most one
    alt_types = 0
    altBase   = "."
    altCount  = 0

    if (nA > 0) { alt_types++; if (alt_types == 1) { altBase = "A"; altCount = nA } }
    if (nC > 0) { alt_types++; if (alt_types == 1) { altBase = "C"; altCount = nC } }
    if (nG > 0) { alt_types++; if (alt_types == 1) { altBase = "G"; altCount = nG } }
    if (nT > 0) { alt_types++; if (alt_types == 1) { altBase = "T"; altCount = nT } }

    if (depth == 0) {
        status = "NONPASS_NO_DATA"
    } else if (alt_types <= 1) {
        status = "PASS"
    } else {
        status = "NONPASS_MULTI"
    }

    # chr pos REF nREF ALT_base nALT depth status
    print chr, pos, REF, nREF, altBase, altCount, depth, status
}
```

Appendice 2: file count_ref_alt.awk

```
#!/usr/bin/awk -f

BEGIN {
    if (e == 0) e = 0.01         # sequencing error rate
    ln10 = log(10)
}
```

```
/^#/ { next }
{
    chr      = $1
    pos      = $2
    REF      = $3
    nREF     = $4 + 0
    ALT      = $5
    nALT     = $6 + 0
    depth    = $7 + 0
    status   = $8

    # only use PASS sites
    if (status !~ /^PASS/) next

    n = nREF + nALT
    if (n == 0) next

    k = nALT   # number of ALT reads

    # binomial model: P(k | n, p)
    # p = P(ALT | genotype)
    # This is exactly the same model we saw in the slides
    pRR = e
    pRA = 0.5
    pAA = 1 - e

    # You probably remember that the binomial also has the binomial coefficient. However,
for each site that is the same for every genotype (it depends only on the number of
alleles of one type and the other!).
    # Thus it simplifies out and we can omit it!!
    lnL_RR = k*log(pRR) + (n-k)*log(1-pRR)
    lnL_RA = k*log(pRA) + (n-k)*log(1-pRA)
    lnL_AA = k*log(pAA) + (n-k)*log(1-pAA)

    # choose best genotype (0/0, 0/1, 1/1)
    lnLmax = lnL_RR
    GT01   = "0/0"
    GTalleles = REF "/" REF

    if (lnL_RA > lnLmax) {
        lnLmax = lnL_RA
        GT01   = "0/1"
        GTalleles = REF "/" ALT
    }
    if (lnL_AA > lnLmax) {
        lnLmax = lnL_AA
        GT01   = "1/1"
        GTalleles = ALT "/" ALT
    }

    # phred-scaled genotype likelihoods relative to best. This is how it is reported in
some VCF files, including those generated via bcftools
    GL_RR = -10 * (lnL_RR - lnLmax) / ln10
    GL_RA = -10 * (lnL_RA - lnLmax) / ln10
    GL_AA = -10 * (lnL_AA - lnLmax) / ln10

    # Output:
    # chr pos REF ALT nREF nALT depth GT01 GTalleles GL_RR GL_RA GL_AA
    printf "%s\t%s\t%s\t%s\t%d\t%d\t%d\t%s\t%s\tGL=%.2f,%.2f,%.2f\n", \
           chr, pos, REF, ALT, nREF, nALT, depth, GT01, GTalleles, \
           GL_RR, GL_RA, GL_AA
}
```