Variant Annotation and Other Exercises

Sean Davis

February 4, 2014

A Example Variant

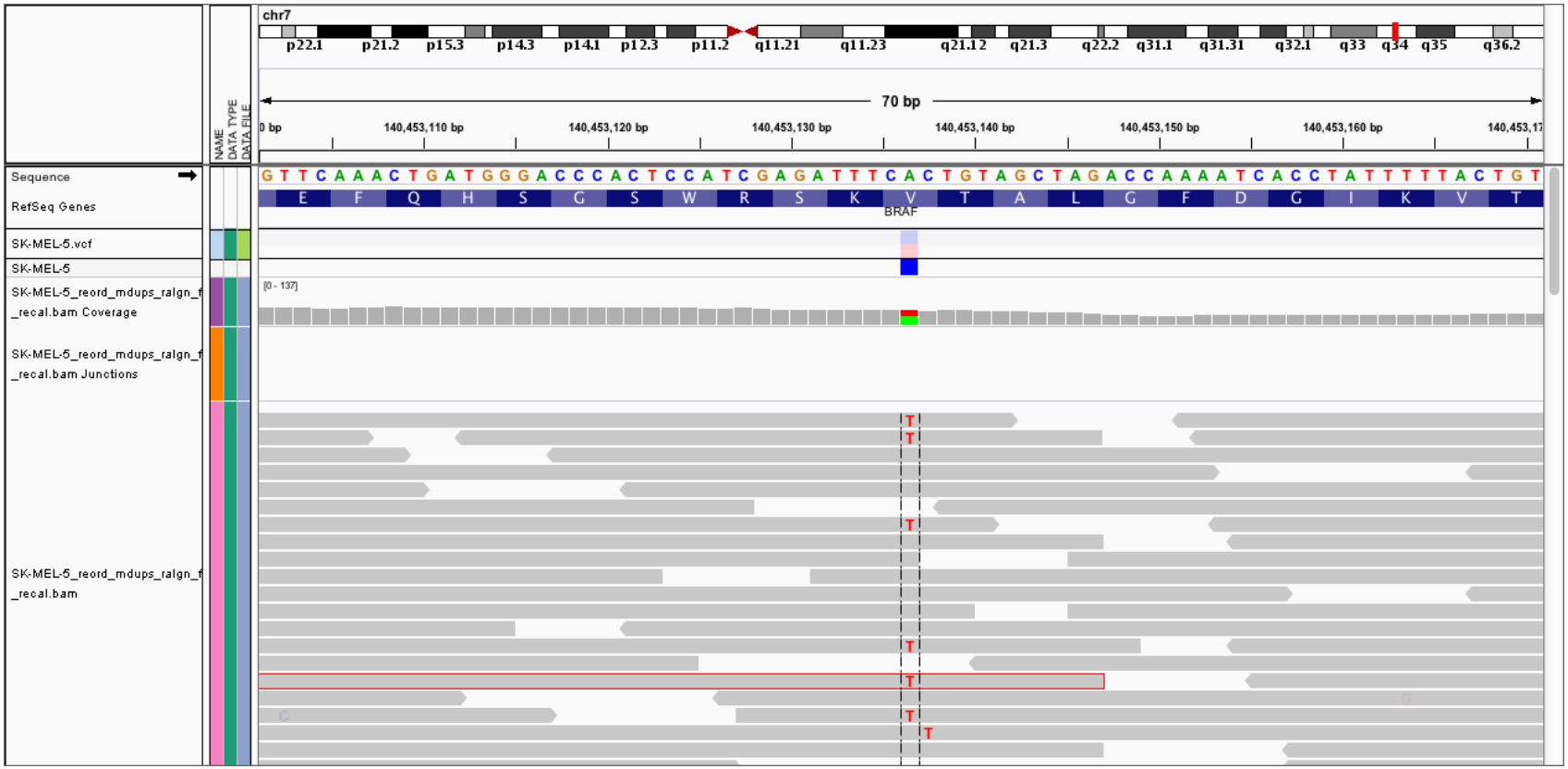


Figure : A true variant in a cancer sample as viewed in the Integrated Genomic Viewer (IGV) software

Tasks

- Read in a VCF file and locate all variants with respect to genes
- Predict the coding effect of the variants in a VCF file
- Investigate mutation spectrum in two cancer samples that have undergone exome sequencing

Possible locations for variants

Location	Details
coding	falls *within* a coding region
fiveUTR	falls *within* a 5' untranslated region
threeUTR	falls *within* a 3' untranslated region
intron	falls *within* an intron region
intergenic	does not fall *within* a transcript associated with a gene
spliceSite	overlaps any portion of the first 2 or last 2 nucleotides of ar
promoter	falls *within* a promoter region of a transcript

Table : Variant locations

Necessary Pieces of Information to Locate Variants

- Gene models and transcripts
- Variants for annotation

Actually Locating Variants

```
library(VariantAnnotation)
fl <- system.file("extdata", "chr22.vcf.gz", package = "VariantAnnotation")
vcf <- readVcf(fl, genome = "hg19")
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
codingvar <- locateVariants(vcf, txdb, CodingVariants())

## Warning:  none of seqlevels(query) match seqlevels(subject)

seqlevels(vcf) = paste0("chr", seqlevels(vcf))
codingvar <- locateVariants(vcf, txdb, CodingVariants())
head(codingvar, 3)
allvar <- locateVariants(vcf, txdb, AllVariants())

## Warning:  trimmed start values to be positive
## Warning:  trimmed end values to be <= seqlengths

head(allvar)
```

Predicting Coding Effects of Variants

- Gene models and transcripts
- Variants for annotation
- *Sequence information*

Actually Predicting Variants

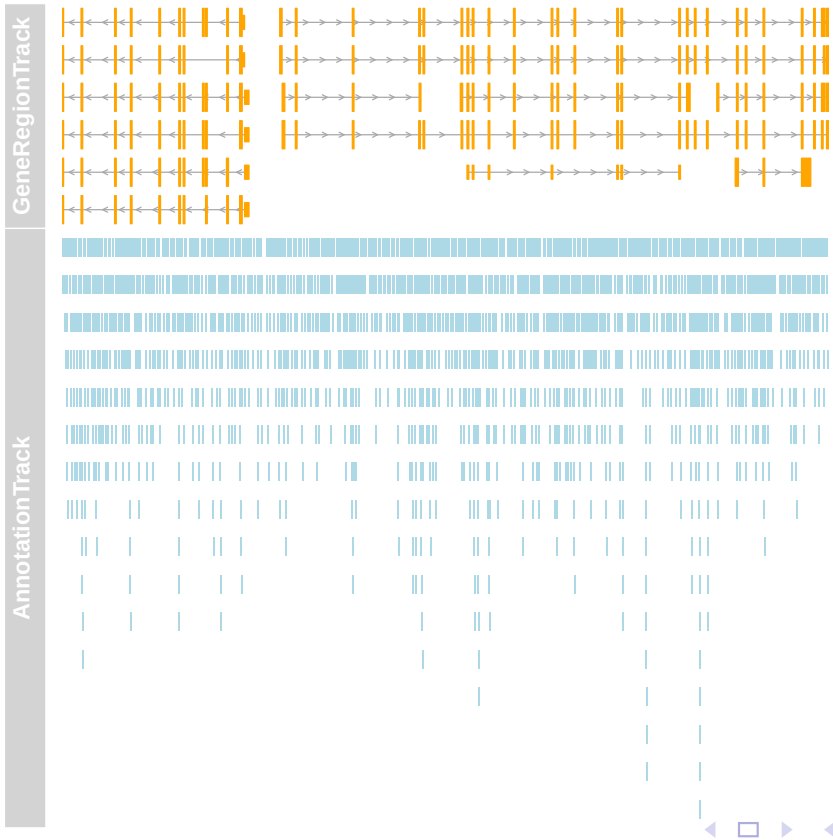
```
library(BSgenome.Hsapiens.UCSC.hg19)
coding <- predictCoding(vcf, txdb, seqSource = Hsapiens)
coding[5:7]
```

When the resulting *varCodon* is not a multiple of 3 it cannot be translated. The consequence is considered a *frameshift* and *varAA* will be missing.

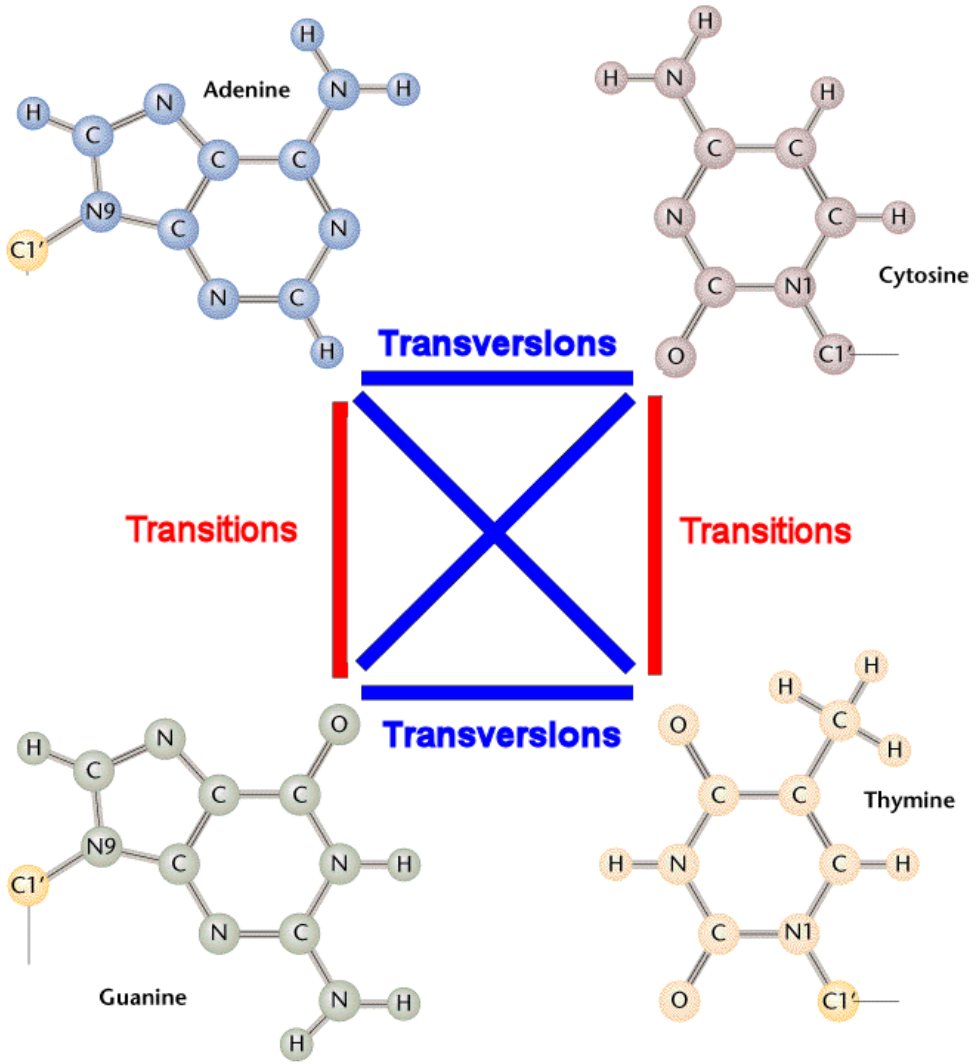
```
## CONSEQUENCE is 'frameshift' where translation is not pos
coding[mcols(coding)$CONSEQUENCE == "frameshift"]
```

Visualizing Variants in Genomic Context Using Gviz

```
library(Gviz)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
geneTrack = GeneRegionTrack(txdb, chromosome = "chr22", start = 5e+07, end = 5.1e+07)
variantTrack = AnnotationTrack(rowData(vcf))
plotTracks(list(geneTrack, variantTrack), from = 50500000, to = 50600000)
```



Transitions and Transversions



Transitions and Transversions

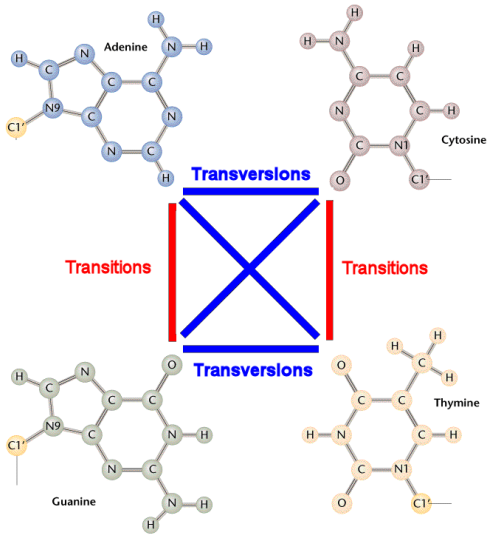
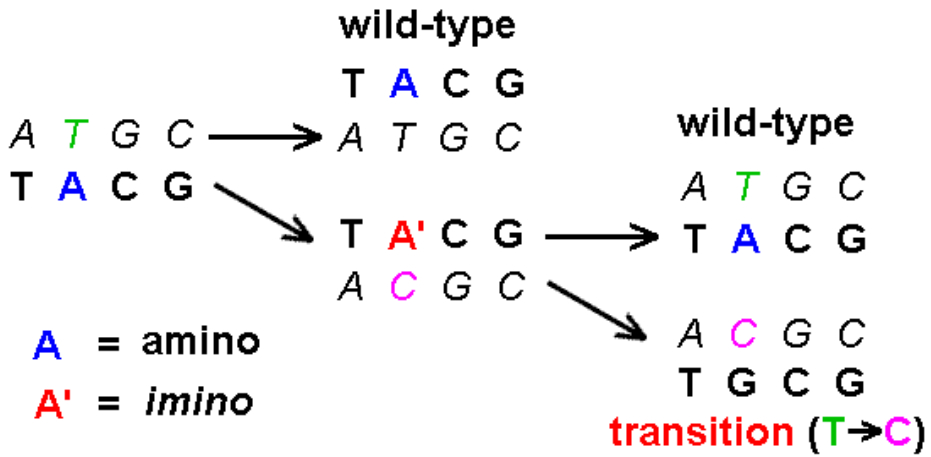


Figure : DNA substitution mutations are of two types. Transitions are interchanges of two-ring purines (A–G) or of one-ring pyrimidines (C–T): they therefore involve bases of similar shape. Transversions are interchanges of purine for pyrimidine bases, which therefore involve exchange of one-ring and two-ring structures. Note that transversions (tv) are twice as likely as transitions(ti), leading to an expected ratio of ti/tv of 0.5.

Spontaneous Mutation Alters ti/tv Ratio



Spontaneous Mutation Alters ti/tv Ratio

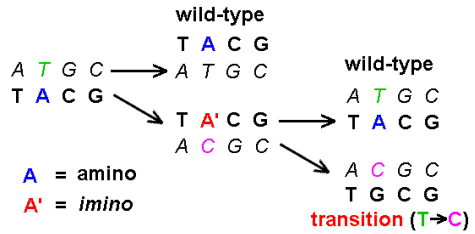


Figure : In the original double-stranded DNA molecule, A in the standard (amino) form pairs with T. During replication, the two strands separate. In the upper diagram, T pairs with A as usual, which replicates the wild-type sequence. In the lower diagram, A has undergone a tautomeric shift to the non-standard (imino) form A', *which pairs with C*. In the next round of replication, the imino A' shifts back to the amino A form, which pairs with T, which again reproduces the wild-type sequence. Replication of the other strand pairs C with G. By comparison with the original molecule, the result is a T–C mutation. A tautomeric shift in one strand has produced a transition mutation in the complementary strand. This process leads to a ti/tv ratio that is typically much larger than the expected 0.5.

ti/tv ratio and false positive variants

- If all observed variants are true positives, then we should observe approximately the ti/tv for the exome, which is reliably estimated at 3.3.
- If all observed variants are false positives, then we should observe the naive expected ti/tv, or 0.5.
- In reality, the observed ti/tv ratio is a mixture of true and false positives.

ti/tv ratio and false positive variants

Given the observed ti/tv ($titv_{obs}$), the false positive ti/tv ($titv_{fp}$), the true positive ti/tv ($titv_{tp}$), and α, the proportion of SNVs that are true positives, we can write:

$$\alpha(titv_{tp}) + (1 - \alpha)(titv_{fp}) = titv_{obs} \quad (1)$$

Solving for 1 - α gives an estimate of the false positive rate.

$$FPR = 1 - \alpha = 1 - (titv_{obs} - titv_{fp})/(titv_{tp} - titv_{fp}) \quad (2)$$

What do we need to find ti/tv

ref \ alt	A	C	G	T
A	0	520	3195	328
C	592	0	770	3598
G	3782	756	0	568
T	359	3349	542	0

Investigating Cancer Sample Tasks

- Load two VCF files representing a cancer sample
- Define a function to produce a data frame of SNPs with two columns:
  - Reference allele
  - Alternate allele (the variant)
- Define a function to calculate the ti/tv ratio for a vcf file
- Subset our two VCF files to include only the "Type 2" variants
- Calculate the ti/tv ratio for each sample
- Make a plot of the mutation spectrum for each sample type

Examining our VCF file

```
# you may need to change the file name to load in the data
vcfMel = readVcf("../data/SK-MEL-5.vcf.gz", genome = "hg19")
vcfLung = readVcf("../data/A549_ATCC.vcf.gz", genome = "hg19")
nrow(vcfMel)
nrow(vcfLung)
vcfMel
vcfLung
rowData(vcfMel)
info(vcfMel)
rowData(vcfMel)
header(vcfMel)
ref(vcfMel)
alt(vcfMel)
```

Our functions

Our functions

```
vcf2snpDF = function(vcf) {
    refall = as.character(ref(vcf))
    altall = as.character(unlist(alt(vcf))[start(PartitioningByEnd(alt(vcf)))])
    tmpDF = data.frame(ref = refall, alt = altall, stringsAsFactors = FALSE)
    # snps only
    tmpDF = tmpDF[nchar(tmpDF$ref) == 1 & nchar(tmpDF$alt) == 1, ]
    return(tmpDF)
}
```

Our functions

```
vcf2snpDF = function(vcf) {
    refall = as.character(ref(vcf))
    altall = as.character(unlist(alt(vcf))[start(PartitioningByEnd(alt(vcf)))])
    tmpDF = data.frame(ref = refall, alt = altall, stringsAsFactors = FALSE)
    # snps only
    tmpDF = tmpDF[nchar(tmpDF$ref) == 1 & nchar(tmpDF$alt) == 1, ]
    return(tmpDF)
}
```

```
titv = function(vcf) {
    variantDF = vcf2snpDF(vcf)
    tbl = table(variantDF)
    ti = tbl["C", "T"] + tbl["T", "C"] + tbl["A", "G"] + tbl["G", "A"]
    tv = tbl["A", "C"] + tbl["C", "A"] + tbl["A", "T"] + tbl["T", "A"] + tbl["C
        "G"] + tbl["G", "C"] + tbl["G", "T"] + tbl["T", "G"]
    return(list(ti = ti, tv = tv, tbl = tbl, titv = ti/tv))
}
```

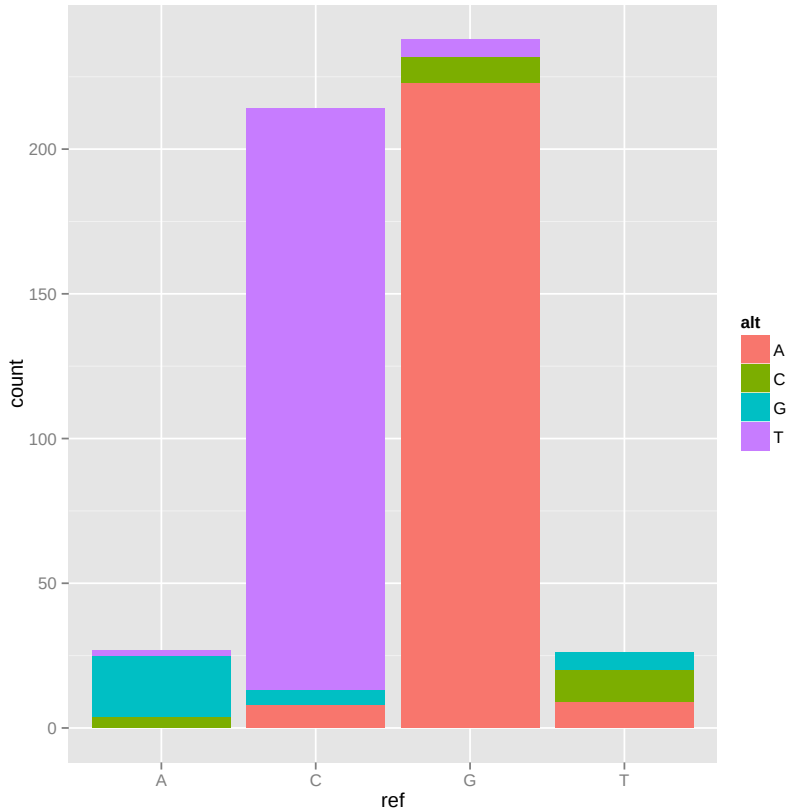
Subset to include only "Type 2" variants

```
vcfMelT2 = vcfMel[grep("Type2", rowData(vcfMel)$FILTER)]
vcfLungT2 = vcfLung[grep("Type2", rowData(vcfLung)$FILTER)]
```

```
titv(vcfMelT2)
titv(vcfLungT2)
```

And a plot of the mutation spectrum

```
library(ggplot2)
ggplot(vcf2snpDF(vcfMelT2), aes(x = ref, fill = alt)) + geom_bar()
```




```
ggplot(vcf2snpDF(vcfLungT2), aes(x = ref, fill = alt)) + geom_bar()
```

Remnants of DNA Damage in Cancer Cells

In lung cancer, polycyclic aromatic hydrocarbons (PAH) cause an increase in transversions (G-T and C-A), leading to a decrease in ti/tv.

In melanoma, thymine dimers associated with UV damage lead to increased transitions, leading to an increase in ti/tv.