

B allele frequencies at the SNPs sharing a segment with the bridge mutations, comparing Met1 and Met2.

Contents

1	Preparation	1
1.1	Allele depth at heterozygous sites	1
1.1.1	Functions	1
1.1.2	Data	1
1.2	Loading segmentation	2
1.3	Positions of bridge mutations	2
1.4	Single-cell allele depth at the bridge mutations	3
1.5	Division of cells into met1 and met2	3
1.5.1	Functions	3
1.5.2	Data	4
1.6	Dividing cells	4
2	Plots	4
2.1	Functions	4
2.2	CHN1	5
2.3	FHIT	9
2.4	APC	16
2.5	ATP7B	20

1 Preparation

1.1 Allele depth at heterozygous sites

1.1.1 Functions

```
> library(tidyverse)
> library(stringr)
> unpack.ad <- Vectorize(function(str)
+ {
+   # Initialize output as a four-element vector of 0's
+   output <- as.integer(rep(0,4))
+   # If input is a missing value, output stays as c(0,0,0,0)
+   # Otherwise, each has allele depth, in order of appearance in the file
+   # Which iirc has reference first? Need to check this
+   if (!is.na(str))
+   {
+     nums <- as.integer(str_split(str, ",")[[1]])
+     output[1:length(nums)] <- nums
+   }
+ }
```

```
+   return(output)
+ })
```

1.1.2 Data

```
> # At sites where the combined data from bulk normal liver and bulk
> # normal colon had at least 10 reads for each of two different nucleotides,
> # what is the allele depth in single cells?
> co8.hetsites.ad <- read_tsv("C08_hetsites_ad.txt")
> co8.hetsites.ad.cells <- co8.hetsites.ad[,-(1:13)]
> # 3 dimensional matrix of allele depth
> #co8.hetsites.ad.cells.unpacked <- array(dim=c(dim(co8.hetsites.ad.cells), 4))
> #for (i in 1:nrow(co8.hetsites.ad.cells))
> #{
> #   for (j in 1:ncol(co8.hetsites.ad.cells))
> #   {
> #       co8.hetsites.ad.cells.unpacked[i,j,] <- unpack.ad(co8.hetsites.ad.cells[i,j])
> #   }
> #}
> #dimnames(co8.hetsites.ad.cells.unpacked) <-
> #   list(rownames(co8.hetsites.ad.cells),
> #         colnames(co8.hetsites.ad.cells),
> #         NULL)
> load("co8_hetsites_ad_cells_unpacked.Rdata")
```

1.2 Loading segmentation

```
> chrom.file <- read.table("../data/marco_bam/my.genome")
> chrom.abspos.end <- cumsum(as.numeric(chrom.file[,2]))
> names(chrom.abspos.end) <- chrom.file[,1]
> chrom.abspos.start <- c(0,chrom.abspos.end[1:(length(chrom.abspos.end)-1)])
> names(chrom.abspos.start) <- names(chrom.abspos.end)
> # Contribution of the chromosome number to absolute position
> chr.contribution <- sapply(as.character(co8.hetsites.ad$CHROM),
+                             function(x) chrom.abspos.start[x])
> # Absolute position of heterozygous snps
> snp.abspos <- co8.hetsites.ad$POS + chr.contribution
> ## ANNOTATE VARIANTS BY SEGMENT FROM COPY NUMBER DATA
> # Loading segment boundaries
> seg.data <- read.delim('../colon_cancer_cn/seg_data/C08.3ormore.rle.txt')
> chr.names <- c(sprintf("chr%d",1:22), "chrX", "chrY")
> seg.boundaries <- Reduce(rbind,
+   lapply(1:24, function(chr.num) with(seg.data,
+     data.frame(chromosome=chr.names[chr.num],
+       start.pos=c(start.pos[chrom==chr.num][1], end.pos[chrom==chr.num]))
+   ))
```

```

+ )
> # Converting this to a labeling of each variant by segment
> seg.number.within.chrom <- unlist(lapply(chr.names,
+   function(chr.name) as.integer(cut(
+     co8.hetsites.ad$POS[co8.hetsites.ad$CHROM==chr.name],
+     seg.boundaries$start[seg.boundaries$chromosome==chr.name]))))
> seg.name <- as.factor(ifelse(is.na(seg.number.within.chrom),
+   NA, sprintf("%s:%d", co8.hetsites.ad$CHROM, seg.number.within.chrom)))
> reference.seg.number.within.chrom <- unlist(lapply(1:24,
+   function(i) 1:sum(seg.data$chrom==i)))
> ordered.levels <- sprintf("%s:%d",
+   chr.names[seg.data$chrom],
+   reference.seg.number.within.chrom)
> seg.number <- sapply(seg.name,
+   function(sn) ifelse(is.na(sn), NA, which(ordered.levels==sn)))

```

1.3 Positions of bridge mutations

- CHN1_chr2_175779934
- FHIT_chr3_60412480
- APC_chr5_112175328
- ATP7B_chr13_52534322

```

> chn1.chrom <- 2
> chn1.pos <- 175779934
> fhit.chrom <- 3
> fhit.pos <- 60412480
> apc.chrom <- 5
> apc.pos <- 112175328
> atp7b.chrom <- 13
> atp7b.pos <- 52534322
> chn1.seg <- with(seg.data,
+   which(chrom==2 & start.pos < 175779934 & end.pos > 175779934))
> fhit.seg <- with(seg.data,
+   which(chrom==3 & start.pos < 60412480 & end.pos > 60412480))
> apc.seg <- with(seg.data,
+   which(chrom==5 & start.pos < 112175328 & end.pos > 112175328))
> atp7b.seg <- with(seg.data,
+   which(chrom==13 & start.pos < 52534322 & end.pos > 52534322))

```

1.4 Single-cell allele depth at the bridge mutations

```

> library(data.table)
> #bridge.mutations <- c("CHN1_chr2_175779934", "FHIT_chr3_60412480",

```

```

> # "ATP7B_chr13_52534322", "APC_chr5_112175328")
> #bridge.chrom <- str_extract(bridge.mutations, "chr\\d+")
> #bridge.pos <- as.integer(str_extract(bridge.mutations, "\\d+$"))
> #exome.ad.full <- fread("../data/C08/exome_AD.txt", verbose=FALSE)
> ##is.cell <- grep("^\\w+-\\d+\\.AD", colnames(exome.ad.full))
> #is.cell <- grep("^\\w+_\\d+$", colnames(exome.ad.full))
> ##cell.name.matches <- str_match(colnames(exome.ad.full)[is.cell], "^((\\w+)-(\\d+)\\.AD$)")
> #cell.name.matches <- str_match(colnames(exome.ad.full)[is.cell], "^((\\w+)_ (\\d+)$)")
> #colnames(exome.ad.full)[is.cell] <- sprintf("%s_%s", cell.name.matches[,2], cell.name.matches[,3])
> #bridge.ad.full <- with(exome.ad.full, exome.ad.full[CHROM %in% bridge.chrom & POS %in% bridge.pos])
> #bridge.ad.full <- as.data.frame(bridge.ad.full)
> #rownames(bridge.ad.full) <- c("CHN1", "FHIT", "APC", "ATP7B")
> #save(bridge.ad.full, file="bridge_ad_full.Rdata")
> load("bridge_ad_full.Rdata")

```

1.5 Division of cells into met1 and met2

1.5.1 Functions

```

> library(sna)
> descendents <- function(mut)
+ {
+   descendent.nodes <- colnames(distmat)[!is.infinite(distmat[mut,])]
+   descendent.cells <- descendent.nodes[descendent.nodes %in% cellnames]
+   return(descendent.cells)
+ }

```

1.5.2 Data

Loading in the SCITE trees, from which the classification of cells into Met1 and Met2 subclones will be derived.

```

> dot.file <- readLines("C08_m10.renamed.gv")
> scite.tree <- read.dot(textConnection(str_replace_all(
+   dot.file, '[ ;]', '')))
> distmat <- geodist(scite.tree, count.paths=FALSE)$gdist
> colnames(distmat) <- colnames(scite.tree)
> rownames(distmat) <- rownames(scite.tree)
> cellnames <- colnames(scite.tree)[grep("^([MP] [DA])", colnames(scite.tree))]

```

1.6 Dividing cells

Using the SCITE trees to make vectors of cell names in the Met1 and Met2 subclones; and making a list of cell names in the diploid population based on which cells were drawn from the diploid DAPI peak during flow sorting, as indicated by a "D" in their cell names.

```

> met1 <- descendants("LING02_chr9_29123273")
> met1.ad.names <- sprintf("%s.AD", str_replace(met1, "_", "-"))
> met1.hetsites.ad <- co8.hetsites.ad.cells.unpacked[,met1.ad.names,]
> met2 <- descendants("NR4A3_chr9_102595561")
> met2.ad.names <- sprintf("%s.AD", str_replace(met2, "_", "-"))
> met2.hetsites.ad <- co8.hetsites.ad.cells.unpacked[,met2.ad.names,]
> diploid <- colnames(scite.tree)[grep("[MP][D]", colnames(scite.tree))]
> diploid.ad.names <- sprintf("%s.AD", str_replace(diploid, "_", "-"))
> diploid.ad.names <- sprintf("%s.AD", str_replace(diploid, "_", "-"))
> diploid.hetsites.ad <- co8.hetsites.ad.cells.unpacked[,diploid.ad.names,]

```

2 Plots

2.1 Functions

```

> first.segs.of.chrom.indices <- c(1,which(diff(seg.data$chrom)==1)+1)
> seg.bin.number <- c(0,cumsum(seg.data$n.probes)[-nrow(seg.data)])+1
> first.segs.of.chrom.bin.numbers <- seg.bin.number[first.segs.of.chrom.indices]
> cn.name.parts <- str_match(colnames(seg.data)[-1:5], "C08\\.(\\[MP\\]\\w+\\.\\.\\.\\d+)")
> standardized.cn.names <- apply(cn.name.parts[,c(2,3)], 1,
+                               function(s) sprintf("%s_%s", s[1], s[2]))
> # Copied from last figure:
> met2.cn <- c("MA_52",
+ "MA_59",
+ "MA_61",
+ "MA_54",
+ "MA_66",
+ "MA_56",
+ "MA_65",
+ "MA_67",
+ "MA_62",
+ "MA_71")
> met1.cn <- c("MA_55",
+ "MA_64",
+ "MA_50",
+ "MA_57",
+ "MA_72",
+ "MA_63",
+ "MA_49",
+ "MA_51")
> met1.cn.indices <- which(standardized.cn.names %in% met1.cn)
> met2.cn.indices <- which(standardized.cn.names %in% met2.cn)
> view.cn <- function(m,...)
+   image(t(t(m)[nrow(t(m)):1,]), axes=TRUE,
+         col=colorRampPalette(c("blue","white","red"))(100), ...)

```

```

> expand <- function(cell.prof)
+ {
+   rep(cell.prof, seg.data$n.probes)
+ }
> show.selection <- function(segs)
+ {
+   end.bins <- cumsum(seg.data$n.probes)
+   start.bins <- c(0,end.bins[-length(end.bins)])+1
+   tot.bins <- sum(seg.data$n.probes)
+   midpoints <- (start.bins+end.bins)/2
+   view.cn(atan(scale(exp(cbind(apply(seg.data[,-(1:5)][,met2.cn.indices], 2, expand),
+   apply(seg.data[,-(1:5)][,met1.cn.indices], 2, expand))))))
+   text(first.segs.of.chrom.bin.numbers/tot.bins, c(0,.1),1:24, cex=.5, col='black')
+   abline(v=c(start.bins[segs],end.bins[segs])/tot.bins, col='green', lwd=1)
+   text(midpoints[segs]/tot.bins, c(.9,1), segs, col='green')
+ }

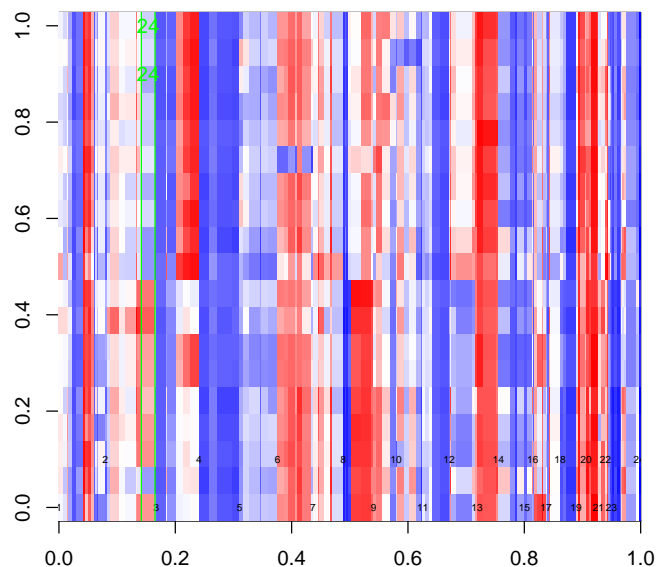
```

2.2 CHN1

```

> show.selection(chn1.seg)

```



```

> chn1.seg.pos <- seg.number==chn1.seg & !is.na(seg.number)
> met1.hetsites.chn1 <- met1.hetsites.ad[chn1.seg.pos,,]

```

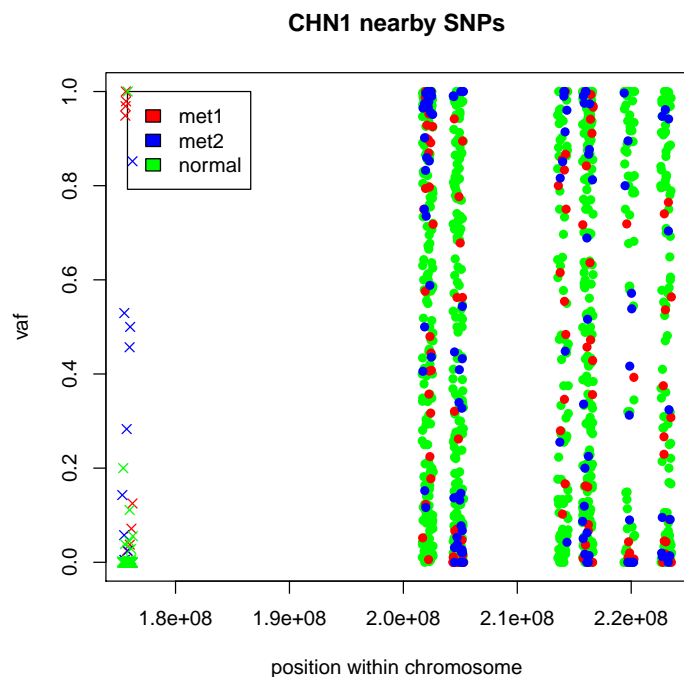
```

> met1.total.depth.chn1 <- apply(met1.hetsites.chn1, c(1,2), sum)
> for (i in 1:4)
+ {
+   met1.hetsites.chn1[,i][met1.total.depth.chn1 < 10] <- NA
+ }
> met2.hetsites.chn1 <- met2.hetsites.ad[chn1.seg.pos,,]
> met2.total.depth.chn1 <- apply(met2.hetsites.chn1, c(1,2), sum)
> for (i in 1:4)
+ {
+   met2.hetsites.chn1[,i][met2.total.depth.chn1 < 10] <- NA
+ }
> diploid.hetsites.chn1 <- diploid.hetsites.ad[chn1.seg.pos,,]
> diploid.total.depth.chn1 <- apply(diploid.hetsites.chn1, c(1,2), sum)
> for (i in 1:4)
+ {
+   diploid.hetsites.chn1[,i][diploid.total.depth.chn1 < 10] <- NA
+ }
> hetsite.positions <- co8.hetsites.ad$POS[chn1.seg.pos]
> chn1.mut.vaf <- unpack.ad(bridge.ad.full["CHN1",-(1:13)])[2,]/apply(unpack.ad(bridge.ad.full["CHN1",-(1:13)]), 2, FUN=function(x) sum(x)/length(x)))

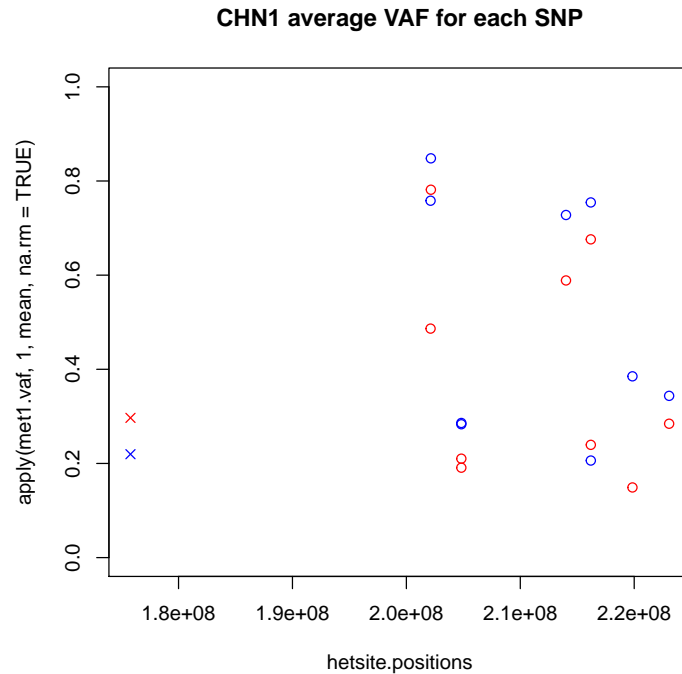
> set.seed(836)
> # Plotting the SNPs
> xrange <- diff(range(c(chn1.pos, hetsite.positions)))
> jdiv <- 100
> #plot(hetsite.positions, rep(0,length(hetsite.positions)), col='white', ylim=c(min(c(met1.hetsites.chn1[,1,2]),met2.hetsites.chn1[,1,2]),max(c(met1.hetsites.chn1[,1,2]),met2.hetsites.chn1[,1,2])), col='white', ylim=c(min(c(met1.hetsites.chn1[,1,2]),met2.hetsites.chn1[,1,2]),max(c(met1.hetsites.chn1[,1,2]),met2.hetsites.chn1[,1,2])),
> plot(c(chn1.pos, hetsite.positions), rep(0,length(hetsite.positions)+1), col='white', ylim=c(min(c(met1.hetsites.chn1[,1,2]),met2.hetsites.chn1[,1,2]),max(c(met1.hetsites.chn1[,1,2]),met2.hetsites.chn1[,1,2])),
+   main="CHN1 nearby SNPs", xlab="position within chromosome", ylab="vaf")
> for (i in 1:ncol(diploid.hetsites.chn1))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         diploid.hetsites.chn1[,i,2]/apply(diploid.hetsites.chn1[,i,],1,sum), col='green', pch=16)
+ }
> for (i in 1:ncol(met1.hetsites.chn1))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met1.hetsites.chn1[,i,2]/apply(met1.hetsites.chn1[,i,],1,sum), pch=16, col='red')
+ }
> for (i in 1:ncol(met2.hetsites.chn1))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met2.hetsites.chn1[,i,2]/apply(met2.hetsites.chn1[,i,],1,sum), col='blue', pch=16)
+ }
> legend(min(c(chn1.pos, hetsite.positions)), 1, c("met1", "met2", "normal"), c("red", "blue", "green"), col=c("red", "blue", "green"), pch=c(16, 16, 16))
> # Plotting the mutations themselves
> points(jitter(rep(chn1.pos, length(met1)), amount=xrange/jdiv), chn1.mut.vaf[met1], col='red', pch=16)
> points(jitter(rep(chn1.pos, length(met2)), amount=xrange/jdiv), chn1.mut.vaf[met2], col='blue', pch=16)

```

```
> points(jitter(rep(chn1.pos, length(diploid))), amount=xrange/jdiv), chn1.mut.vaf[diploid],
```



```
> met1.vaf <- met1.hetsites.chn1[, ,2] / apply(met1.hetsites.chn1, c(1,2), sum)
> met2.vaf <- met2.hetsites.chn1[, ,2] / apply(met2.hetsites.chn1, c(1,2), sum)
> plot(hetsite.positions,
+       apply(met1.vaf, 1, mean, na.rm=TRUE),
+       ylim=c(0,1), xlim=range(c(chn1.pos, hetsite.positions)), col='red',
+       main="CHN1 average VAF for each SNP")
> points(hetsite.positions,
+       apply(met2.vaf, 1, mean, na.rm=TRUE),
+       ylim=c(0,1), col='blue')
> points(chn1.pos, mean(chn1.mut.vaf[met1], na.rm=TRUE), col='red', pch=4)
> points(chn1.pos, mean(chn1.mut.vaf[met2], na.rm=TRUE), col='blue', pch=4)
```

```
> minority <- function(p) pmin(p, 1-p)
> mean(minority(apply(met1.vaf, 1, mean, na.rm=TRUE)))

[1] 0.2793628

> mean(minority(apply(met2.vaf, 1, mean, na.rm=TRUE)))

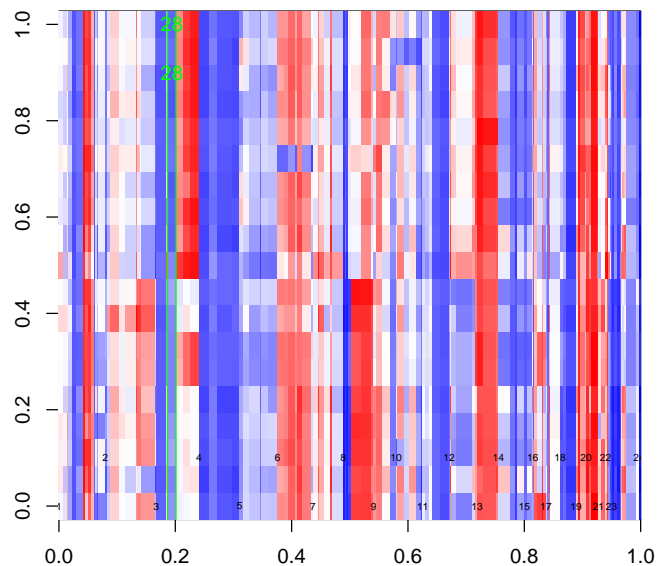
[1] 0.2683958

> diploid.vaf <- diploid.hetsites.chn1[,2] / apply(diploid.hetsites.chn1, c(1,2), sum)
> mean(minority(apply(diploid.vaf, 1, mean, na.rm=TRUE)))

[1] 0.445891
```

2.3 FHIT

```
> show.selection(fhit.seg)
```



```

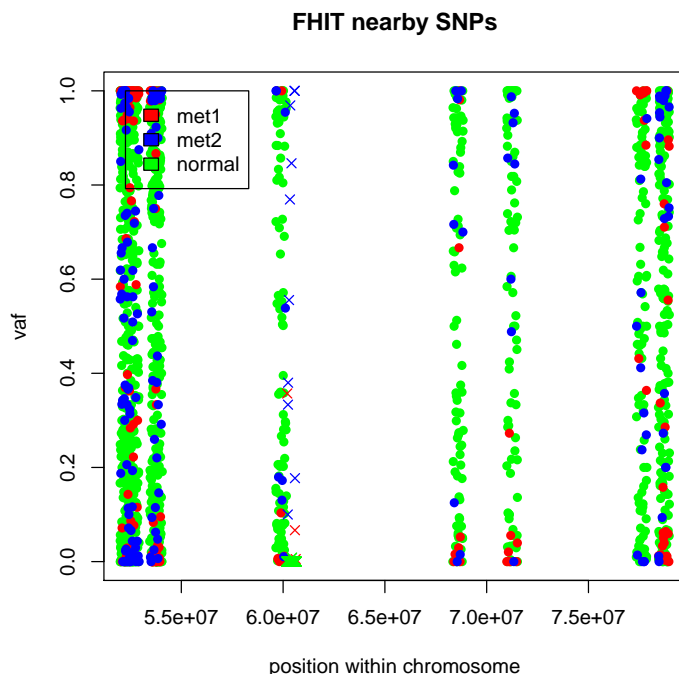
> fhit.seg.pos <- seg.number==fhit.seg & !is.na(seg.number)
> met1.hetsites.fhit <- met1.hetsites.ad[fhit.seg.pos,,]
> met1.total.depth.fhit <- apply(met1.hetsites.fhit, c(1,2), sum)
> for (i in 1:4)
+ {
+   met1.hetsites.fhit[,i][met1.total.depth.fhit < 10] <- NA
+ }
> met2.hetsites.fhit <- met2.hetsites.ad[fhit.seg.pos,,]
> met2.total.depth.fhit <- apply(met2.hetsites.fhit, c(1,2), sum)
> for (i in 1:4)
+ {
+   met2.hetsites.fhit[,i][met2.total.depth.fhit < 10] <- NA
+ }
> diploid.hetsites.fhit <- diploid.hetsites.ad[fhit.seg.pos,,]
> diploid.total.depth.fhit <- apply(diploid.hetsites.fhit, c(1,2), sum)
> for (i in 1:4)
+ {
+   diploid.hetsites.fhit[,i][diploid.total.depth.fhit < 10] <- NA
+ }
> hetsite.positions <- co8.hetsites.ad$POS[fhit.seg.pos]
> fhit.mut.vaf <- unpack.ad(bridge.ad.full["FHIT",-(1:13)])[2,]/apply(unpack.ad(bridge.ad.full["FHIT",-(1:13)]), 2, FUN=function(x) sum(x==1))

```

```

> set.seed(836)
> # Plotting the SNPs
> xrange <- diff(range(c(fhit.pos, hetsite.positions)))
> jdiv <- 100
> #plot(hetsite.positions, rep(0,length(hetsite.positions)), col='white', ylim=c(min(c(met1.
> plot(c(fhit.pos, hetsite.positions), rep(0,length(hetsite.positions)+1), col='white', ylim
+     main="FHIT nearby SNPs", xlab="position within chromosome", ylab="vaf")
> for (i in 1:ncol(diploid.hetsites.fhit))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         diploid.hetsites.fhit[,i,2]/apply(diploid.hetsites.fhit[,i,],1,sum), col='green',
+   }
> for (i in 1:ncol(met1.hetsites.fhit))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met1.hetsites.fhit[,i,2]/apply(met1.hetsites.fhit[,i,],1,sum), pch=16, col='red')
+ }
> for (i in 1:ncol(met2.hetsites.fhit))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met2.hetsites.fhit[,i,2]/apply(met2.hetsites.fhit[,i,],1,sum), col='blue', pch=16)
+ }
> legend(min(c(fhit.pos, hetsite.positions)), 1, c("met1", "met2", "normal"), c("red", "blue", "black"))
> # Plotting the mutations themselves
> points(jitter(rep(fhit.pos, length(met1)), amount=xrange/jdiv), fhit.mut.vaf[met1], col='red')
> points(jitter(rep(fhit.pos, length(met2)), amount=xrange/jdiv), fhit.mut.vaf[met2], col='blue')
> points(jitter(rep(fhit.pos, length(diploid)), amount=xrange/jdiv), fhit.mut.vaf[diploid], col='black')

```



The SNP right next to it looks kind of suspicious. What's going on there?

```
> suspicious.site.index <- which.min(abs(hetsite.positions - fhit.pos))
> suspicious.site.met1.vaf <- met1.hetsites.fhit[suspicious.site.index,,2] / apply(met1.hetsites.fhit, 1, FUN = function(x) sum(x == 1))
> suspicious.site.met2.vaf <- met2.hetsites.fhit[suspicious.site.index,,2] / apply(met2.hetsites.fhit, 1, FUN = function(x) sum(x == 1))
> suspicious.site.met1.vaf
```

MA-91.AD	MA-37.AD	MA-45.AD	MA-88.AD	MA-94.AD	MA-40.AD
0.000000000	0.006711409	0.000000000	1.000000000	1.000000000	0.103448276
MA-33.AD	MA-35.AD	MA-86.AD	MA-29.AD	MA-90.AD	MA-27.AD
0.011494253	0.004601227	0.000000000	NA	NA	NA
MA-48.AD	MA-28.AD	MA-85.AD			
NA	NA	0.000000000			

```
> suspicious.site.met2.vaf
```

MA-32.AD	MA-87.AD	MA-89.AD	MA-31.AD	MA-34.AD	MA-47.AD	MA-95.AD
NA	0.53846154	0.17948718	0.01036269	0.13043478	1.00000000	NA
MA-46.AD	MA-93.AD	MA-30.AD	MA-36.AD	MA-38.AD	MA-43.AD	
0.17241379	NA	NA	0.95454545	NA	1.00000000	

```
> wilcox.test(suspicious.site.met1.vaf, suspicious.site.met2.vaf)
```

Wilcoxon rank sum test with continuity correction

```
data: suspicious.site.met1.vaf and suspicious.site.met2.vaf
W = 16, p-value = 0.03487
alternative hypothesis: true location shift is not equal to 0
```

There's a statistically significant difference at $\alpha = 0.05$, but that's probably just because I picked this one out for looking weird. I bet it's nothing like the difference you see in the mutation itself:

```
> fhit.mut.vaf[met1]

      MA_91      MA_37      MA_45      MA_88      MA_94      MA_40
0.000000000 0.000000000      NaN 0.000000000 0.357142857 0.000000000
      MA_33      MA_35      MA_86      MA_29      MA_90      MA_27
      NaN 0.000000000 0.066666667 0.000000000 0.005376344 0.000000000
      MA_48      MA_28      MA_85
0.000000000 0.000000000 0.000000000

> fhit.mut.vaf[met2]

      MA_32      MA_87      MA_89      MA_31      MA_34      MA_47      MA_95      MA_46
1.0000000 0.3800000 0.1000000 0.0000000 0.0000000 0.1772152 0.5555556 0.8461538
      MA_93      MA_30      MA_36      MA_38      MA_43
0.7692308 0.3333333 0.9689922 0.0000000 1.0000000

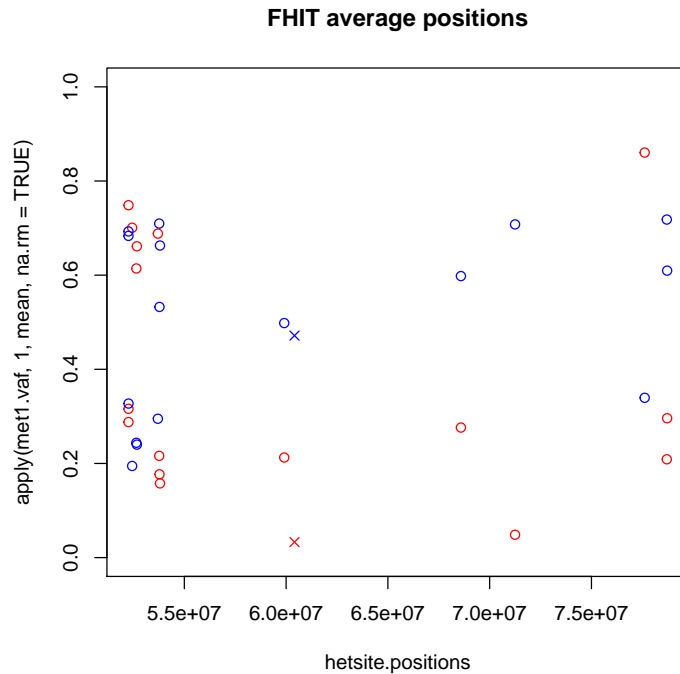
> wilcox.test(fhit.mut.vaf[met1],fhit.mut.vaf[met2])
```

Wilcoxon rank sum test with continuity correction

```
data: fhit.mut.vaf[met1] and fhit.mut.vaf[met2]
W = 27, p-value = 0.001781
alternative hypothesis: true location shift is not equal to 0
```

Yup. So I'm not going to treat this SNP differently, just going to average it in with everything else.

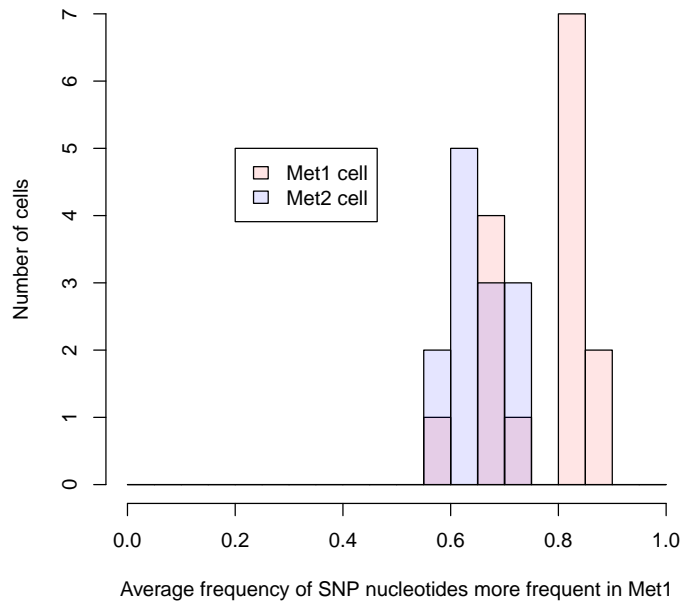
```
> met1.vaf <- met1.hetsites.fhit[,2] / apply(met1.hetsites.fhit, c(1,2), sum)
> met2.vaf <- met2.hetsites.fhit[,2] / apply(met2.hetsites.fhit, c(1,2), sum)
> diploid.vaf <- diploid.hetsites.fhit[,2] / apply(diploid.hetsites.fhit, c(1,2), sum)
> plot(hetsite.positions,
+      apply(met1.vaf, 1, mean, na.rm=TRUE),
+      ylim=c(0,1), xlim=range(c(fhit.pos, hetsite.positions)), col='red',
+      main="FHIT average positions")
> points(hetsite.positions,
+      apply(met2.vaf, 1, mean, na.rm=TRUE),
+      ylim=c(0,1), col='blue')
> points(fhit.pos, mean(fhit.mut.vaf[met1],na.rm=TRUE), col='red', pch=4)
> points(fhit.pos, mean(fhit.mut.vaf[met2],na.rm=TRUE), col='blue', pch=4)
```



These look really different. What's *that* mean?

What's going on here? Are there SNPs that are more frequent in every Met1 cell? Or are there a subset of Met1 cells which have them, or a subset of Met2 that lack them?

```
> # TRUE for SNPs that are variant at higher frequency in Met1
> met1.haplotype <- minority(apply(met1.vaf, 1, mean, na.rm=TRUE)) > minority(apply(met2.vaf, 1, mean, na.rm=TRUE))
> # Mean VAF of the met1 haplotype in each cell
> met1.metiness <- apply(met1.vaf, 2,
+   function(cell.vaf) mean(ifelse(met1.haplotype, cell.vaf, 1-met1.vaf), na.rm=TRUE))
> met2.metiness <- apply(met2.vaf, 2,
+   function(cell.vaf) mean(ifelse(met1.haplotype, cell.vaf, 1-met1.vaf), na.rm=TRUE))
> hist(met1.metiness, breaks=seq(0,1,.05), col=rgb(1,0,0,.1),
+   xlab="Average frequency of SNP nucleotides more frequent in Met1",
+   ylab="Number of cells", main="")
> hist(met2.metiness, breaks=seq(0,1,.05), col=rgb(0,0,1,.1),add=TRUE)
> legend(.2, 5, c("Met1 cell", "Met2 cell"), c(rgb(1,0,0,.1), rgb(0,0,1,.1)))
```

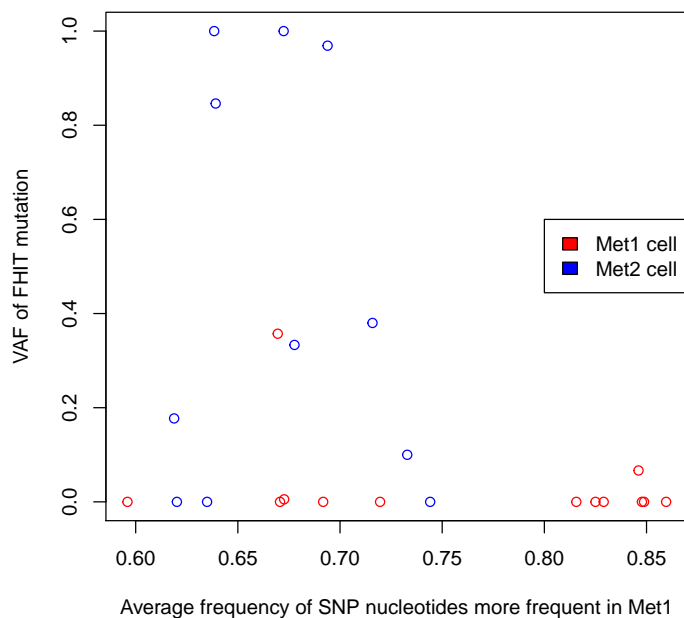


The answer is, there's a subset of Met1 cells that have a higher frequency of these SNPs. So, maybe about half of the Met1 cells have an extra copy of that site. That corresponds to observation of the single-cell copy number data, which shows that this is the area where a fraction of Met1 cells have an amplification from copy number 3 to 4. That would explain this, though the VAFs don't look right for three-fourths. Met1 lacks the FHIT mutation. So, the cells with the unusual SNP frequencies shouldn't have the FHIT mutation, and it shouldn't really affect anything.

One possibility, though, is that FHIT just has a lower detection rate in Met1 cells because of an amplification of the non-mutated FHIT allele. In that case, in the Met1 cells *without* the amplification, we should see just as many FHIT reads as in Met2.

```
> # simplify ad names
> san <- function(ad.names)
+ {
+   apply(str_match(ad.names, "^((\\w+)-(\\d+))\\.AD$")[,2:3], 1,
+         function(v) paste(v, collapse="_"))
+ }
> plot(met1.met1ness, fh1t.mut.vaf[san(names(met1.met1ness))],
+      col='red', ylim=c(0,1),
+      xlab="Average frequency of SNP nucleotides more frequent in Met1",
+      ylab="VAF of FHIT mutation")
```

```
> points(met2.met1ness, fhit.mut.vaf[san(names(met2.met1ness))], col='blue')
> legend(.8, .6, c("Met1 cell", "Met2 cell"), c("red", "blue"))
```



No, that's not the pattern. Even in the cluster with the low values of the deletion, Met1 tends not to have it. Final conclusion is that the difference in SNP frequencies between these two subclones is due to the copy number change between the Met1 and Met2 subclones. That's going to produce a difference in mean minority VAF between the two subclones, but not a large enough drop to indicate loss of a chromosome containing the FHIT mutation.

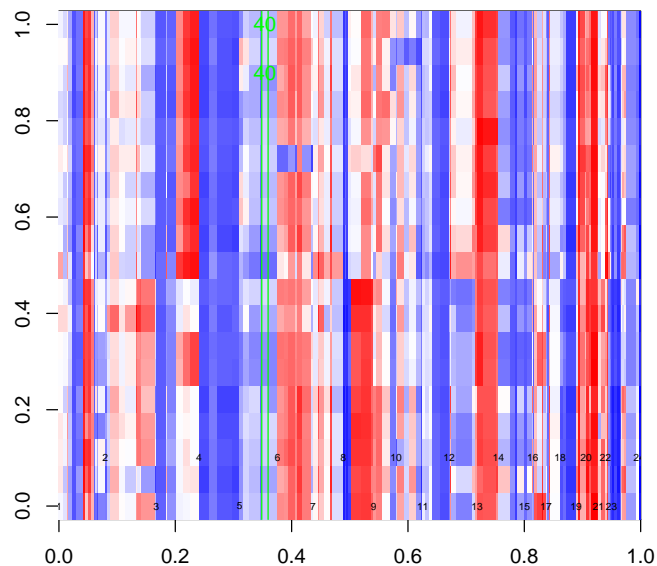
```
> mean(minority(apply(met1.vaf, 1, mean, na.rm=TRUE)))
[1] 0.2452202

> mean(minority(apply(met2.vaf, 1, mean, na.rm=TRUE)))
[1] 0.3264786

> mean(minority(apply(diploid.vaf, 1, mean, na.rm=TRUE)))
[1] 0.4629368
```

2.4 APC

```
> show.selection(apc.seg)
```

```

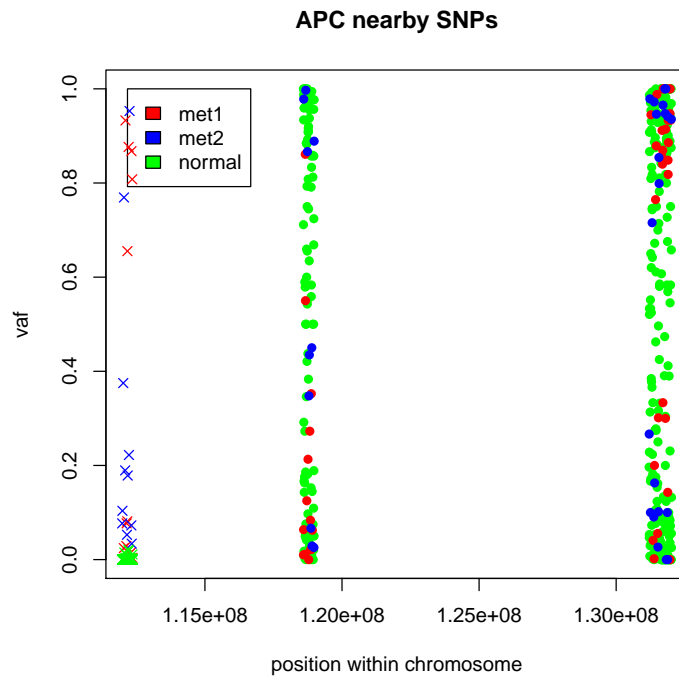
> apc.seg.pos <- seg.number==apc.seg & !is.na(seg.number)
> met1.hetsites.apc <- met1.hetsites.ad[apc.seg.pos,,]
> met1.total.depth.apc <- apply(met1.hetsites.apc, c(1,2), sum)
> for (i in 1:4)
+ {
+   met1.hetsites.apc[,i][met1.total.depth.apc < 10] <- NA
+ }
> met2.hetsites.apc <- met2.hetsites.ad[apc.seg.pos,,]
> met2.total.depth.apc <- apply(met2.hetsites.apc, c(1,2), sum)
> for (i in 1:4)
+ {
+   met2.hetsites.apc[,i][met2.total.depth.apc < 10] <- NA
+ }
> diploid.hetsites.apc <- diploid.hetsites.ad[apc.seg.pos,,]
> diploid.total.depth.apc <- apply(diploid.hetsites.apc, c(1,2), sum)
> for (i in 1:4)
+ {
+   diploid.hetsites.apc[,i][diploid.total.depth.apc < 10] <- NA
+ }
> hetsite.positions <- co8.hetsites.ad$POS[apc.seg.pos]
> apc.mut.vaf <- unpack.ad(bridge.ad.full["APC",- (1:13)]) [2,]/apply(unpack.ad(bridge.ad.full

```

```

> set.seed(836)
> # Plotting the SNPs
> xrange <- diff(range(c(apc.pos, hetsite.positions)))
> jdiv <- 100
> #plot(hetsite.positions, rep(0,length(hetsite.positions)), col='white', ylim=c(min(c(met1.
> plot(c(apc.pos, hetsite.positions), rep(0,length(hetsite.positions)+1), col='white', ylim=
+     main="APC nearby SNPs", xlab="position within chromosome", ylab="vaf")
> for (i in 1:ncol(diploid.hetsites.apc))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         diploid.hetsites.apc[,i,2]/apply(diploid.hetsites.apc[,i,],1,sum), col='green', p
+ }
> for (i in 1:ncol(met1.hetsites.apc))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met1.hetsites.apc[,i,2]/apply(met1.hetsites.apc[,i,],1,sum), pch=16, col='red')
+ }
> for (i in 1:ncol(met2.hetsites.apc))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met2.hetsites.apc[,i,2]/apply(met2.hetsites.apc[,i,],1,sum), col='blue', pch=16)
+ }
> legend(min(c(apc.pos, hetsite.positions)), 1, c("met1", "met2", "normal"), c("red", "blue")
> # Plotting the mutations themselves
> points(jitter(rep(apc.pos, length(met1)), amount=xrange/jdiv), apc.mut.vaf[met1], col='red')
> points(jitter(rep(apc.pos, length(met2)), amount=xrange/jdiv), apc.mut.vaf[met2], col='blue')
> points(jitter(rep(apc.pos, length(diploid)), amount=xrange/jdiv), apc.mut.vaf[diploid], col='green')

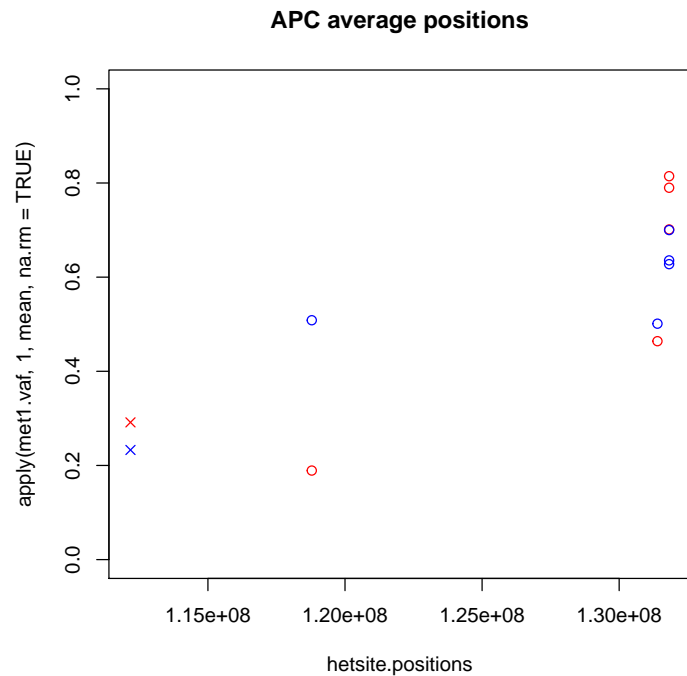
```



```

> met1.vaf <- met1.hetsites.apc[,2] / apply(met1.hetsites.apc, c(1,2), sum)
> met2.vaf <- met2.hetsites.apc[,2] / apply(met2.hetsites.apc, c(1,2), sum)
> plot(hetsite.positions,
+      apply(met1.vaf, 1, mean, na.rm=TRUE),
+      ylim=c(0,1), xlim=range(c(apc.pos, hetsite.positions)), col='red',
+      main="APC average positions")
> points(hetsite.positions,
+      apply(met2.vaf, 1, mean, na.rm=TRUE),
+      ylim=c(0,1), col='blue')
> points(apc.pos, mean(apc.mut.vaf[met1], na.rm=TRUE), col='red', pch=4)
> points(apc.pos, mean(apc.mut.vaf[met2], na.rm=TRUE), col='blue', pch=4)

```



```
> mean(minority(apply(met1.vaf, 1, mean, na.rm=TRUE)))
```

```
[1] 0.269489
```

```
> mean(minority(apply(met2.vaf, 1, mean, na.rm=TRUE)))
```

```
[1] 0.4054403
```

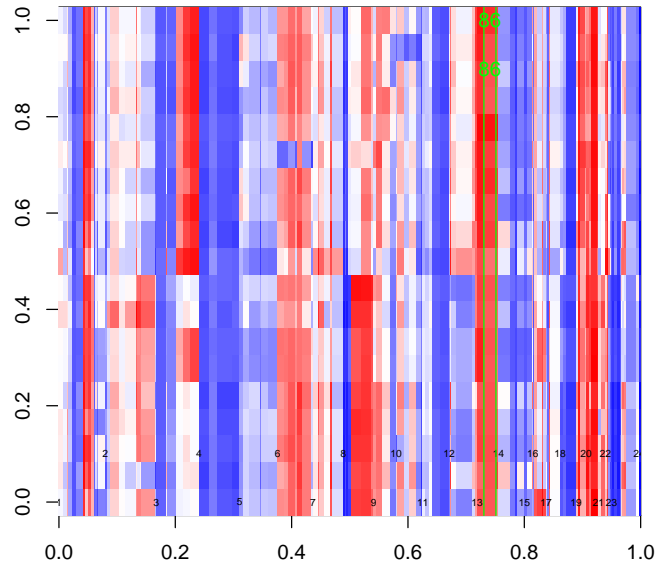
```
> diploid.vaf <- diploid.hetsites.apc[,2] / apply(diploid.hetsites.apc, c(1,2), sum)
```

```
> mean(minority(apply(diploid.vaf, 1, mean, na.rm=TRUE)))
```

```
[1] 0.4669824
```

2.5 ATP7B

```
> show.selection(atp7b.seg)
```



```

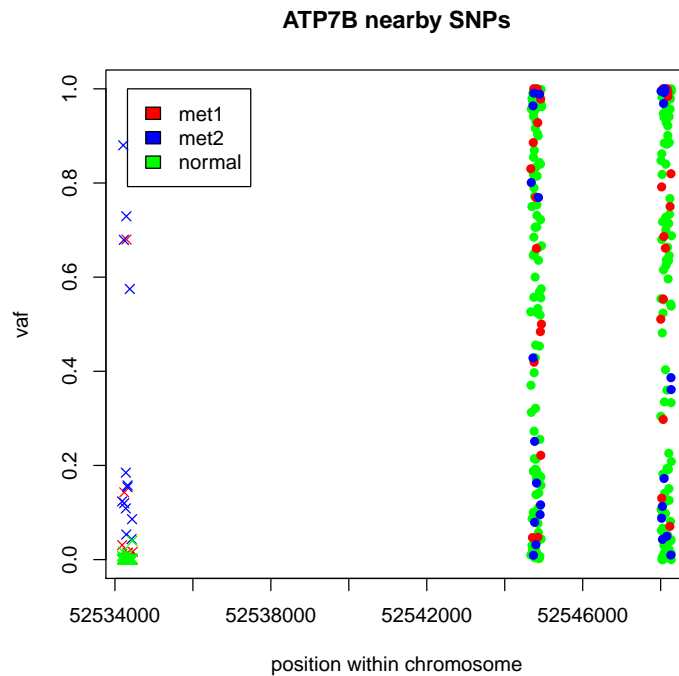
> atp7b.seg.pos <- seg.number==atp7b.seg & !is.na(seg.number)
> met1.hetsites.atp7b <- met1.hetsites.ad[atp7b.seg.pos,,]
> met1.total.depth.atp7b <- apply(met1.hetsites.atp7b, c(1,2), sum)
> for (i in 1:4)
+ {
+   met1.hetsites.atp7b[,i][met1.total.depth.atp7b < 10] <- NA
+ }
> met2.hetsites.atp7b <- met2.hetsites.ad[atp7b.seg.pos,,]
> met2.total.depth.atp7b <- apply(met2.hetsites.atp7b, c(1,2), sum)
> for (i in 1:4)
+ {
+   met2.hetsites.atp7b[,i][met2.total.depth.atp7b < 10] <- NA
+ }
> diploid.hetsites.atp7b <- diploid.hetsites.ad[atp7b.seg.pos,,]
> diploid.total.depth.atp7b <- apply(diploid.hetsites.atp7b, c(1,2), sum)
> for (i in 1:4)
+ {
+   diploid.hetsites.atp7b[,i][diploid.total.depth.atp7b < 10] <- NA
+ }
> hetsite.positions <- co8.hetsites.ad$POS[atp7b.seg.pos]
> atp7b.mut.vaf <- unpack.ad(bridge.ad.full["ATP7B",-(1:13)])[2,]/apply(unpack.ad(bridge.ad.

```

```

> set.seed(836)
> # Plotting the SNPs
> xrange <- diff(range(c(atp7b.pos, hetsite.positions)))
> jdiv <- 100
> #plot(hetsite.positions, rep(0,length(hetsite.positions)), col='white', ylim=c(min(c(met1.
> plot(c(atp7b.pos, hetsite.positions), rep(0,length(hetsite.positions)+1), col='white', ylim=
+     main="ATP7B nearby SNPs", xlab="position within chromosome", ylab="vaf")
> for (i in 1:ncol(diploid.hetsites.atp7b))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         diploid.hetsites.atp7b[,i,2]/apply(diploid.hetsites.atp7b[,i,],1,sum), col='green')
+ }
> for (i in 1:ncol(met1.hetsites.atp7b))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met1.hetsites.atp7b[,i,2]/apply(met1.hetsites.atp7b[,i,],1,sum), pch=16, col='red')
+ }
> for (i in 1:ncol(met2.hetsites.atp7b))
+ {
+   points(jitter(hetsite.positions, amount=xrange/jdiv),
+         met2.hetsites.atp7b[,i,2]/apply(met2.hetsites.atp7b[,i,],1,sum), col='blue', pch=16)
+ }
> legend(min(c(atp7b.pos, hetsite.positions)), 1, c("met1", "met2", "normal"), c("red", "blue", "black"))
> # Plotting the mutations themselves
> points(jitter(rep(atp7b.pos, length(met1)), amount=xrange/jdiv), atp7b.mut.vaf[met1], col='red')
> points(jitter(rep(atp7b.pos, length(met2)), amount=xrange/jdiv), atp7b.mut.vaf[met2], col='blue')
> points(jitter(rep(atp7b.pos, length(diploid)), amount=xrange/jdiv), atp7b.mut.vaf[diploid], col='black')

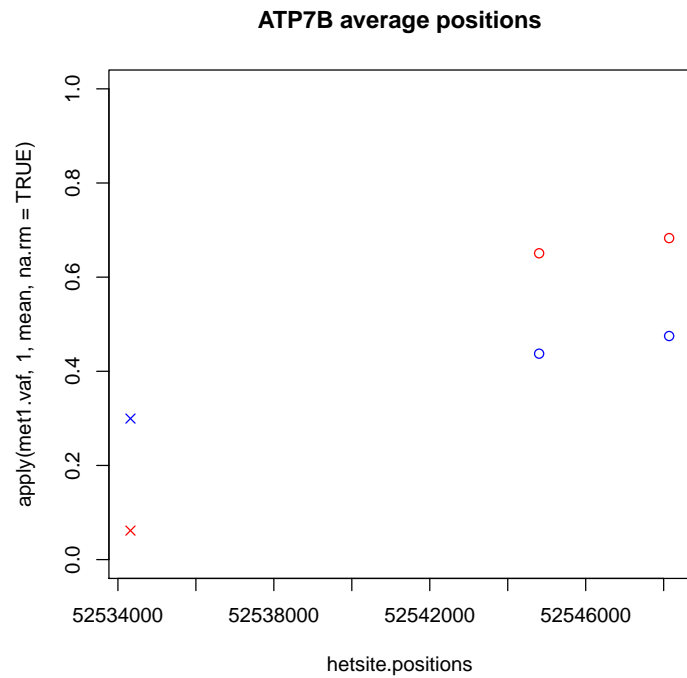
```



```

> met1.vaf <- met1.hetsites.atp7b[,2] / apply(met1.hetsites.atp7b, c(1,2), sum)
> met2.vaf <- met2.hetsites.atp7b[,2] / apply(met2.hetsites.atp7b, c(1,2), sum)
> plot(hetsite.positions,
+      apply(met1.vaf, 1, mean, na.rm=TRUE),
+      ylim=c(0,1), xlim=range(c(atp7b.pos, hetsite.positions)), col='red',
+      main="ATP7B average positions")
> points(hetsite.positions,
+      apply(met2.vaf, 1, mean, na.rm=TRUE),
+      ylim=c(0,1), col='blue')
> points(atp7b.pos, mean(atp7b.mut.vaf[met1],na.rm=TRUE), col='red', pch=4)
> points(atp7b.pos, mean(atp7b.mut.vaf[met2],na.rm=TRUE), col='blue', pch=4)

```



```
> mean(minority(apply(met1.vaf, 1, mean, na.rm=TRUE)))
```

```
[1] 0.3331639
```

```
> mean(minority(apply(met2.vaf, 1, mean, na.rm=TRUE)))
```

```
[1] 0.4561941
```

```
> diploid.vaf <- diploid.hetsites.atp7b[,2] / apply(diploid.hetsites.atp7b, c(1,2), sum)
```

```
> mean(minority(apply(diploid.vaf, 1, mean, na.rm=TRUE)))
```

```
[1] 0.4886251
```