

# Marking errors

Alexander Davis

February 24, 2017

## Contents

<b>1</b>	<b>Code to load data matrices and generate theoretical genotype matrices</b>	<b>1</b>
<b>2</b>	<b>Visualizing the genotype matrices</b>	<b>3</b>
<b>3</b>	<b>Checking correctness of genotype matrices</b>	<b>4</b>
<b>4</b>	<b>Overlaying data matrices and genotype matrices to obtain false positives and false negatives</b>	<b>5</b>
<b>5</b>	<b>Making a figure</b>	<b>7</b>
<b>6</b>	<b>Estimates of error rates</b>	<b>10</b>
6.1	False negative rate . . . . .	10
6.1.1	CO5 . . . . .	11
6.1.2	CO8 . . . . .	11
6.2	Rate of mutation not seen . . . . .	11
6.2.1	CO5 . . . . .	11
6.2.2	CO8 . . . . .	11
6.3	False positive rate . . . . .	11
6.3.1	CO5 . . . . .	11
6.3.2	CO8 . . . . .	12
6.4	False discovery rate . . . . .	12
6.4.1	CO5 . . . . .	12
6.4.2	CO8 . . . . .	12

## 1 Code to load data matrices and generate theoretical genotype matrices

```
> library(sna)
> library(stringr)
> library(igraph)
```

```

> library(gplots)
> # Load data from Marco
> co5.master.mat <- read.table('.././data/C05/C05.master_matrix.txt')
> co8.master.mat <- read.table('.././data/C08/C08.master_matrix.txt')
> # Remove population columns
> co5.data.mat <- co5.master.mat[, -(1:6)]
> co8.data.mat <- co8.master.mat[, -(1:5)]
> # Load trees
> read.dot.tree <- function(path)
+ {
+   dot.file <- readLines(path)
+   read.dot(textConnection(str_replace_all(
+     dot.file, '[ ;]', '')))
+ }
> co5.adjmat <- read.dot.tree("C05_m10.renamed.gv")
> co8.adjmat <- read.dot.tree("C08_m10.renamed.gv")
> co5.tree <- graph_from_adjacency_matrix(co5.adjmat)
> co8.tree <- graph_from_adjacency_matrix(co8.adjmat)
> # Make theoretical genotype matrix
> genotype.matrix <- function(adjmat, tree)
+ {
+   # Figure out which nodes are root, cells, and mutations
+   root.index <- which(colnames(adjmat)=="Root")
+   stopifnot(length(root.index)==1)
+   cell.indices <- grep("[MP][A-Z0-9]*_\\d*$", colnames(adjmat))
+   mut.indices <- (1:ncol(adjmat))[! 1:ncol(adjmat) %in% c(root.index, cell.indices)]
+   # Find paths from root
+   paths <- shortest_paths(tree, root.index, cell.indices)$vpath
+   # Express paths as the numbers of mutations
+   # (counting first mutation as 1, second as 2, etc)
+   path.interiors <- lapply(paths,
+     function(x) if(length(x)>2) {return(x[2:(length(x)-1)])} else {return(integer(0))})
+   path.mut.numbers <- sapply(path.interiors,
+     function(xs) as.integer(sapply(xs, function(x) which(mut.indices==x)[1])))
+   # Produce genotype matrix
+   # (in each row: a 1 for a mutation in the path, 0 otherwise)
+   gm <- t(sapply(path.mut.numbers,
+     function(p) {row<-rep(0,length(mut.indices)); row[p]<-1; return(row)}))
+   colnames(gm) <- colnames(adjmat)[mut.indices]
+   rownames(gm) <- colnames(adjmat)[cell.indices]
+   return(gm)
+ }
> co5.gm <- genotype.matrix(co5.adjmat, co5.tree)
> co8.gm <- genotype.matrix(co8.adjmat, co8.tree)

```

## 2 Visualizing the genotype matrices

```
> view.mat <- function(mat, ...)
+ {
+   clustered.mat <- mat[hclust(dist(mat))$order, hclust(dist(t(mat)))$order]
+   image(t(clustered.mat[nrow(mat):1,]), axes=FALSE, ...)
+ }
> view.mat(co5.gm)
```



What are those weird cells on the bottom with just a few mutations? Oh, those would be cells up near the neck of the tumor tree. Weird cells.

```
> view.mat(co8.gm)
```



### 3 Checking correctness of genotype matrices

Checking that these are correct. In CO5, TMP4 should be in 9 cells. GATA1 should be in 16 cells, all of which have POU2AF1 and not PM4.

```
> sum(co5.gm[, 'TMP4'])
[1] 9

> sum(co5.gm[, 'GATA1'])
[1] 16

> sum(co5.gm[, 'GATA1']==1 & co5.gm[, 'POU2AF1']==1)
[1] 16

> sum(co5.gm[, 'GATA1']==1 & co5.gm[, 'TMP4']==1)
[1] 0
```

In CO8, ALK is in 9 cells, of which 4 also have SPEN. NR4A3 is in 13 cells, of which 4 also have TSHZ3.

```

> sum(co8.gm[, "ALK_chr2_29416591"])
[1] 9
> sum(co8.gm[, "SPEN_chr1_16202934"])
[1] 4
> sum(co8.gm[, "ALK_chr2_29416591"] & co8.gm[, "SPEN_chr1_16202934"]==1)
[1] 4
> sum(co8.gm[, 'NR4A3_chr9_102595561'])
[1] 13
> sum(co8.gm[, "TSHZ3_chr19_31768568"])
[1] 4
> sum(co8.gm[, 'NR4A3_chr9_102595561']==1 & co8.gm[, "TSHZ3_chr19_31768568"]==1)
[1] 4

```

## 4 Overlaying data matrices and genotype matrices to obtain false positives and false negatives

```

> co5.dm <- t(co5.data.mat)
> rownames(co5.dm) <- str_replace_all(rownames(co5.dm), "[:.-]", "_")
> colnames(co5.dm) <- str_extract(colnames(co5.dm), "~[A-Z0-9]+")
> co5.gm.aligned <- co5.gm[rownames(co5.dm), colnames(co5.dm)]
> co5.dm["PD_41",]

```

POU2AF1	CCNE1	MYH9	TCF7L2	KRAS	APC	TP53	FAT3	ROBO2	TDRP
0.0	0.0	0.0	0.0	0.5	1.0	0.0	0.0	0.0	0.0
EYS	GATA1	RBFOX1	TRRAP	ZNF521	TPM4				
0.0	0.0	0.0	0.0	0.0	0.0				

```

> co5.gm.aligned["PD_41",]

```

POU2AF1	CCNE1	MYH9	TCF7L2	KRAS	APC	TP53	FAT3	ROBO2	TDRP
0	0	0	0	0	1	0	0	0	0
EYS	GATA1	RBFOX1	TRRAP	ZNF521	TPM4				
0	0	0	0	0	0				

```

> which(rownames(co5.dm)=="PD_41")
[1] 87

```

```

> co5.pairs <- outer(1:nrow(co5.dm), 1:ncol(co5.dm),
+   Vectorize(function(i,j) sprintf("%s,%s",co5.dm[i,j],co5.gm.aligned[i,j])))
> colnames(co5.pairs) <- colnames(co5.dm)
> rownames(co5.pairs) <- rownames(co5.dm)
> co5.pairs[87,]

```

POU2AF1	CCNE1	MYH9	TCF7L2	KRAS	APC	TP53	FAT3	ROBO2	TDRP
"0,0"	"0,0"	"0,0"	"0,0"	"0.5,0"	"1,1"	"0,0"	"0,0"	"0,0"	"0,0"
EYS	GATA1	RBFOX1	TRRAP	ZNF521	TPM4				
"0,0"	"0,0"	"0,0"	"0,0"	"0,0"	"0,0"				

```

> co5.pairs["PD_41",]

```

POU2AF1	CCNE1	MYH9	TCF7L2	KRAS	APC	TP53	FAT3	ROBO2	TDRP
"0,0"	"0,0"	"0,0"	"0,0"	"0.5,0"	"1,1"	"0,0"	"0,0"	"0,0"	"0,0"
EYS	GATA1	RBFOX1	TRRAP	ZNF521	TPM4				
"0,0"	"0,0"	"0,0"	"0,0"	"0,0"	"0,0"				

```

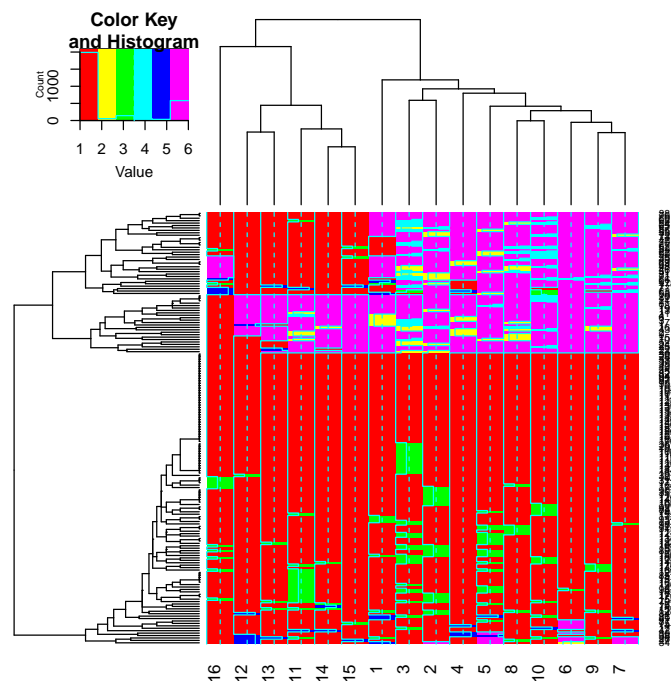
> heatmap.2(matrix(as.numeric(as.factor(co5.pairs)), ncol=ncol(co5.pairs)), col=rainbow(6))
> levels(as.factor(co5.pairs))

```

```

[1] "0,0" "0,1" "0.5,0" "0.5,1" "1,0" "1,1"

```

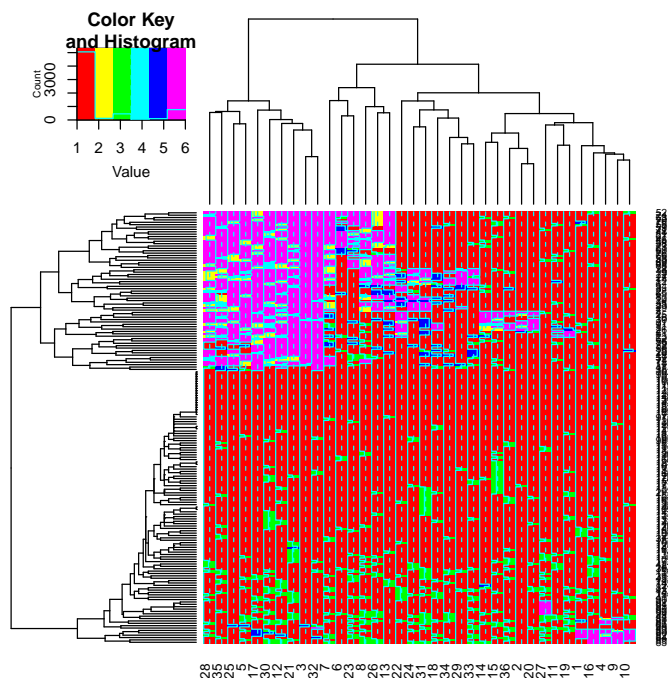


```

> co8.dm <- t(co8.data.mat)
> rownames(co8.dm) <- str_replace_all(rownames(co8.dm), "[:.-]", "_")
> colnames(co8.dm) <- str_replace_all(colnames(co8.dm), "[:.-]", "_")
> co8.gm.aligned <- co8.gm[rownames(co8.dm), colnames(co8.dm)]
> co8.pairs <- outer(1:nrow(co8.dm), 1:ncol(co8.dm),
+   Vectorize(function(i,j) sprintf("%s,%s", co8.dm[i,j], co8.gm.aligned[i,j])))
> colnames(co8.pairs) <- colnames(co8.dm)
> rownames(co8.pairs) <- rownames(co8.dm)
> heatmap.2(matrix(as.numeric(as.factor(co8.pairs))), ncol=ncol(co8.pairs), col=rainbow(6))
> levels(as.factor(co8.pairs))

```

```
[1] "0,0" "0,1" "0.5,0" "0.5,1" "1,0" "1,1"
```



## 5 Making a figure

```

> convert <- function(x) switch(x, "1,0"="FP",
+   "0,1"="FN",
+   "0.5,0"="Missing (no mutation)",
+   "0.5,1"="Missing (mutation)",
+   "1,1"="P",
+   "0,0"="N")

```

```

> co5.errors <- apply(co5.pairs, 2, Vectorize(convert))
> co8.errors <- apply(co8.pairs, 2, Vectorize(convert))

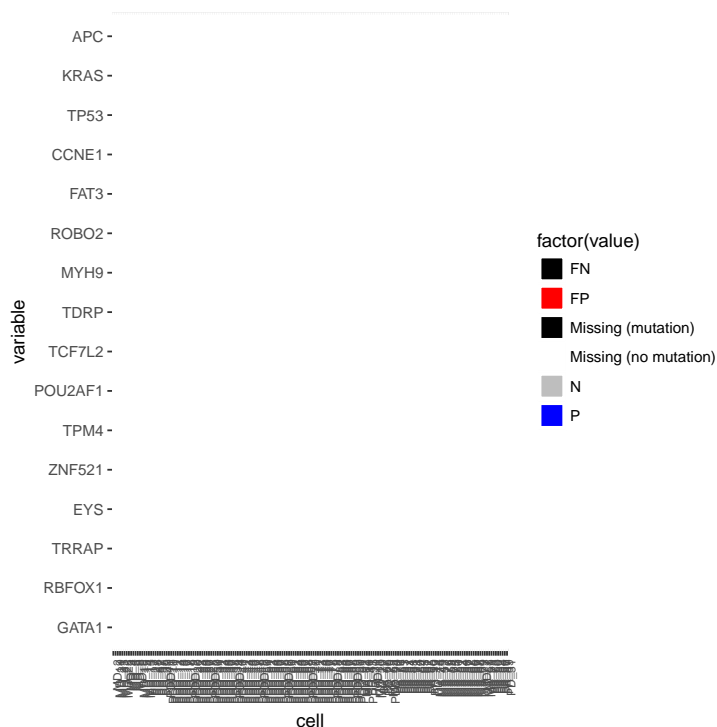
> library(ggplot2)
> library(reshape2)
> order.nodes <- function(adjmat)
+ {
+   names <- c("Root")
+   stack <- NULL
+   while (TRUE)
+   {
+     stack <- c(colnames(adjmat)[which(adjmat[names[1],]!=0)], stack)
+     if (length(stack) == 0) break
+     names <- c(stack[1], names)
+     stack <- stack[-1]
+   }
+   return(names)
+ }
> trim.pos <- function(charvec) str_extract(charvec, "[A-Z0-9]*")
> error.mat <- co8.errors
> gt.mat <- co8.gm
> data.mat <- co8.dm
> adjmat <- co8.adjmat
> plot.error.mat <- function(error.mat, gt.mat, data.mat, adjmat, size=4)
+ {
+   # Add a hierarchical clustering sorting step before sorting by tree
+   mut.hclust.sorted <- colnames(data.mat)[hclust(dist(t(gt.mat)),method='ward.D')$order]
+   cell.hclust.sorted <- rownames(data.mat)[hclust(dist(gt.mat),method='ward.D')$order]
+   adjmat <- adjmat[c("Root",mut.hclust.sorted,cell.hclust.sorted),c("Root",mut.hclust.sorted)]
+   all.order <- order.nodes(adjmat)
+   mut.sorted <- unique(all.order[all.order %in% colnames(error.mat)])
+   cell.sorted <- unique(all.order[all.order %in% rownames(error.mat)])
+   #colnames(error.mat) <- make.unique(trim.pos(colnames(error.mat)))
+   #mut.sorted <- make.unique(trim.pos(mut.sorted))
+   y <- melt(cbind(data.frame(cell=rownames(error.mat)),error.mat), id.vars=c('cell'))
+   ggplot(y, aes(cell, variable)) + geom_tile(aes(fill=factor(value)),color='white',size=1)
+ }
> save.error.mat <- function(error.mat, gt.mat, data.mat, adjmat, filename)
+ {
+   # Add a hierarchical clustering sorting step before sorting by tree
+   mut.hclust.sorted <- colnames(data.mat)[hclust(dist(t(gt.mat)),method='ward.D')$order]
+   cell.hclust.sorted <- rownames(data.mat)[hclust(dist(gt.mat),method='ward.D')$order]
+   adjmat <- adjmat[c("Root",mut.hclust.sorted,cell.hclust.sorted),c("Root",mut.hclust.sorted)]
+   all.order <- order.nodes(adjmat)
+   mut.sorted <- unique(all.order[all.order %in% colnames(error.mat)])
+   cell.sorted <- unique(all.order[all.order %in% rownames(error.mat)])

```



```
+ # Apply the sorting to the matrix
+ sorted.error.mat <- error.mat[cell.sorted,mut.sorted]
+ # Write the matrix to a file
+ write.table(sorted.error.mat, filename, quote=FALSE, sep='\t')
+ }
```

```
> plot.error.mat(co5.errors, co5.gm, co5.dm, co5.adjmat, size=7)
```



```
> png("co5_error_mat.png", width=1514, height=1056)
> plot.error.mat(co5.errors, co5.gm, co5.dm, co5.adjmat, size=7)
> dev.off()
```

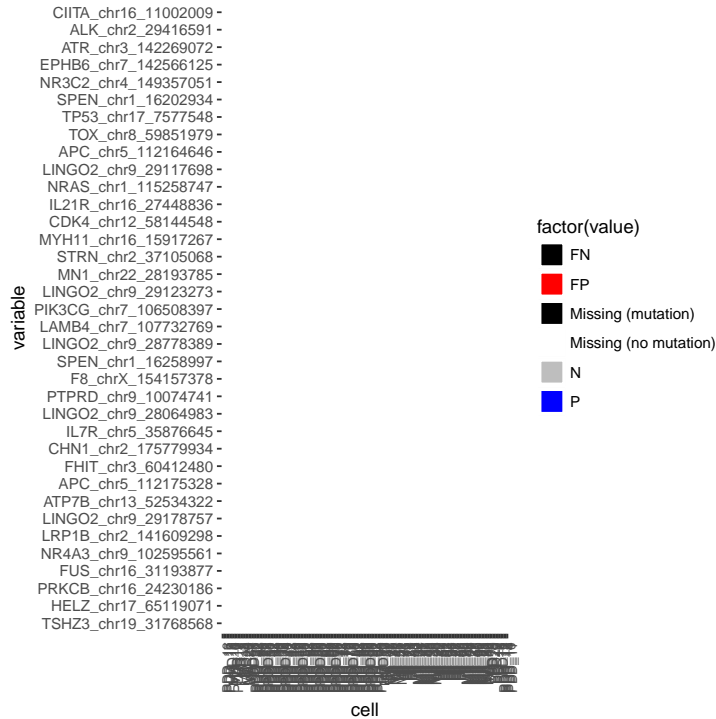
```
null device
      1
```

```
> pdf("co5_error_mat.pdf", width=50, height=30)
> plot.error.mat(co5.errors, co5.gm, co5.dm, co5.adjmat, size=7)
> dev.off()
```

```
null device
      1
```

```
> save.error.mat(co5.errors, co5.gm, co5.dm, co5.adjmat, "co5_error_mat.txt")
```

```
> plot.error.mat(co8.errors, co8.gm, co8.dm, co8.adjmat, size=8)
```



```
> png("co8_error_mat.png", width=1165, height=660)
```

```
> plot.error.mat(co8.errors, co8.gm, co8.dm, co8.adjmat, size=8)
```

```
> dev.off()
```

```
null device
```

```
1
```

```
> pdf("co8_error_mat.pdf", width=50, height=30)
```

```
> plot.error.mat(co8.errors, co8.gm, co8.dm, co8.adjmat,size=8)
```

```
> dev.off()
```

```
null device
```

```
1
```

```
> save.error.mat(co8.errors, co8.gm, co8.dm, co8.adjmat, "co8_error_mat.txt")
```

## 6 Estimates of error rates

### 6.1 False negative rate

Here, I am defining false negative rate as the fraction of mutations that were called as non-mutated. I am *not* including missing sites.

### 6.1.1 CO5

```
> sum(co5.errors=="FN")/  
+ sum(co5.gm==1)  
  
[1] 0.07894737
```

### 6.1.2 CO8

```
> sum(co8.errors=="FN")/  
+ sum(co8.gm==1)  
  
[1] 0.12566
```

## 6.2 Rate of mutation not seen

Here, I consider cases where the mutation was not observed, which includes both sites called as negative, *and* missing values where a mutatio would be theoretically expected.

### 6.2.1 CO5

```
> (sum(co5.errors=='FN') + sum(co5.errors=='Missing (mutation)')) /  
+ sum(co5.gm==1)  
  
[1] 0.1505848
```

### 6.2.2 CO8

```
> (sum(co8.errors=='FN') + sum(co8.errors=='Missing (mutation)')) /  
+ sum(co8.gm==1)  
  
[1] 0.189018
```

## 6.3 False positive rate

Here, I consider mutations called as positive, where no mutation is actually present.

### 6.3.1 CO5

```
> sum(co5.errors=='FP')/  
+ sum(co5.gm==0)  
  
[1] 0.01524954
```

### 6.3.2 CO8

```
> sum(co8.errors=='FP')/  
+ sum(co8.gm==0)
```

```
[1] 0.01748439
```

## 6.4 False discovery rate

Here, I consider the fraction of positive mutation calls which are false.

### 6.4.1 CO5

```
> (sum(co5.errors=='FP')+sum(co5.errors=='P'))
```

```
[1] 614
```

```
> sum(co5.errors=='FP')/  
+ sum(co5.dm==1)
```

```
[1] 0.05374593
```

### 6.4.2 CO8

```
> sum(co8.errors=='FP')/  
+ sum(co8.dm==1)
```

```
[1] 0.113164
```