VDJdb sandbox-1

Load data from Alexey, melt and remove Cys:

```r
library(reshape2)
library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(plyr)
library(MASS)

df <- read.csv("msubstitution_matrix_3_0_0_3_all.csv", stringsAsFactors = F)
df <- melt(df)
```

```
## Using X as id variables
```

```r
df$type <- "outer"

df.1 <- read.csv("substitution_matrix_3_0_0_3_all.csv", stringsAsFactors = F)
df.1 <- melt(df.1)
```

```
## Using X as id variables
```

```r
df.1$type <- "inner"

df <- rbind(df, df.1)

colnames(df) <- c("from", "to", "count", "type")

df <- subset(df, from != "C" & to != "C")
```

Here $C_{ij}^{(s)}$ and $C_{ij}^{(d)}$ amino acid substitution counts for CDR3 alignments with same (s) and distinct (d) antigens. $C_{ii}^{(s,d)}$ represents the number of times amino acid gets unchanged.

Protect against $\log 0$

$$C_{ij}^{(s,d)} \leftarrow C_{ij}^{(s,d)} + 1$$

Probability of not changing amino acid

$$P_{ii}^{(s,d)} = \frac{C_{ii}^{(s,d)}}{\sum_k C_{ik}^{(s,d)}}$$

Probability of $i \to j, i \neq j$ is

$$P_{ij}^{(s,d)} = (1 - P_{ii}^{(s,d)}) \frac{C_{ij}^{(s,d)}}{\sum_{k \neq i} C_{ik}^{(s,d)}}$$

Symmetrize as follows

$$P_{ij}^{(s,d)} = \frac{P_{ij}^{(s,d)}}{P_{ij}^{(s,d)} + P_{ji}^{(s,d)}} P_{ij}^{(s,d)} + \frac{P_{ji}^{(s,d)}}{P_{ij}^{(s,d)} + P_{ji}^{(s,d)}} P_{ji}^{(s,d)}$$

Compute odds ratios as

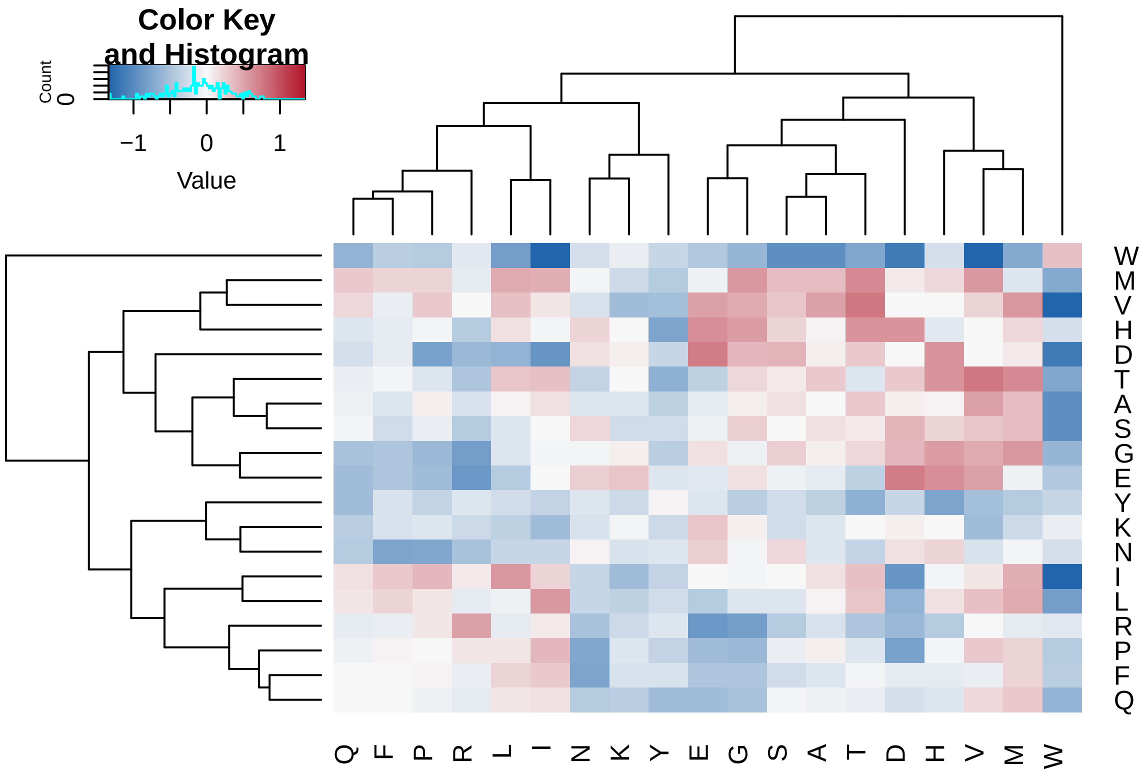
$$Q_{ij} = \log_{10} \frac{P_{ij}^{(s)}}{P_{ij}^{(s)}}$$

Perform calculations, cluster amino acids based on substitution odds ratios

```
df$count <- df$count + 1

# Pii
df <- ddply(df, .(from, type), transform,
            P = count / sum(count))

# Pij
df <- ddply(df, .(from, type), transform,
            P = ifelse(from == to, P,
                       (1 - P[which(from == to)]) * count / sum(count[which(from != to)])))

df.inv <- data.frame(from = df$to, to = df$from, type = df$type, count = df$count, P = df$P)
df <- merge(df, df.inv, by = c("from", "to", "type", "count"))
df$P <- mapply(function(x, y) min(x, y), df$P.x, df$P.y)

df.1 <- ddply(df, .(from, to), summarize, Q = log10(P[which(type=="inner")] / P[which(type=="outer")]))

df.m <- dcast(df.1, from ~ to)
```

```
## Using Q as value column: use value.var to override.
```

```
rownames(df.m) <- df.m$from
df.m$from <- NULL
df.m <- data.matrix(df.m)

df.m <- df.m[order(rownames(df.m)), order(colnames(df.m))]

heatmap.2(df.m, scale="none", symm = T, trace = "none",
          #hclustfun = function(x) hclust(dist(x), method = "ward"),
          col=colorpanel(100, "#2166ac", "#f7f7f7", "#b2182b"))
```

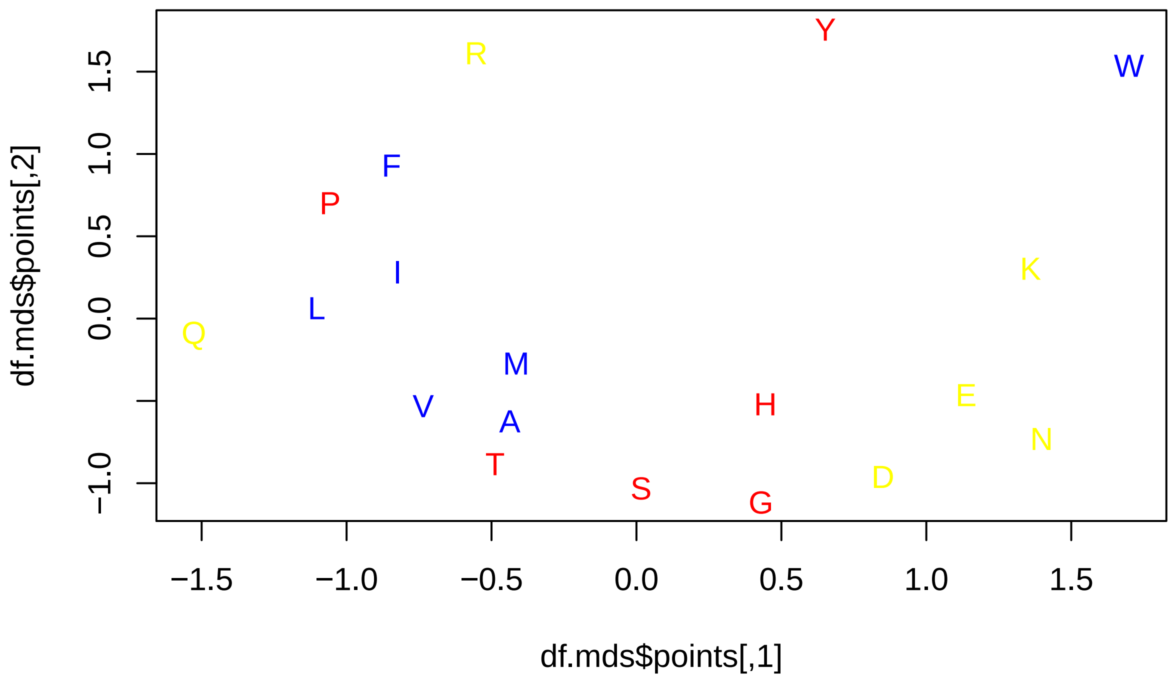


Some MDS analysis

```
# hydrophobic, neutral, hydrophilic
colors <- c(rep("blue", 8), rep("red", 6), rep("yellow", 6))
names(colors) <- strsplit("I V L F C M A W G T S Y P H N D Q E K R", " ")[[1]]

df.mds <- isoMDS(as.dist(-df.m + 2), k = 2)
```

```
## initial  value 23.489940
## iter   5 value 19.227963
## final  value 18.913673
## converged
```

```
plot(df.mds$points, type = "n")
text(df.mds$points, labels = rownames(df.m), col = colors[rownames(df.m)])
```



df.mds$points[,2]
df.mds$points[,1]
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
R
Y
W
F
P
I
L
Q
K
M
V
A
H
E
N
T
S
G
D