Analysis of BRCA2 MAVE Data Set
Bi–Normal, Common Variance Classification Model with Batch Location And Scaling

July 9, 2024

1 High–Level User–Specified Parameters

```
## T degrees of freedom for measurement error model
t.df<-5
## Do you want a test run with a 20% Subset?
subst<-FALSE
## Choose Data Set:
db<-"uvCounts"
## db<-"uvCountsFnl"
##
## Prior Probability Pathogenic, beta distribution parameters:
beta.a<-2.0
beta.b<-8.0  ##mean=0.20, ESS = 10
## Set List of MCMC Control Parameters:
mcmc.pars<-list(iter=10000, ## short run
                burn=5000,
                thin=10)
## mcmc.pars<-list(iter=150000, ## long run
##                  burn=50000,
##                  thin=10)
##mcmc.pars<-list(iter=550000, ## longer run
##                 burn=50000,
##                 thin=25)
```

2 Start Up

Set a seed for the random number generators to ensure repeatabability and load necessary R libraries.

```
## Set Random Seed
set.seed(seed=03122024)
## Load custom functions that will be used below:
source("RFuncs.R",echo=FALSE)
## Libraries:
##library(coda)
```

```
library(rjags)
library(R2WinBUGS)
library(R2jags)
library(mgcv)
```

3 Import Data

Read in the MAVE data and training variant labels:

```
if (db=="uvCounts"){
  uvDB<-read.delim("combined.raw.20.tsv")
}
if (db=="uvCountsFnl"){
  uvDB<-read.delim("combined.raw.tsv")
}
dim(uvDB)

## [1] 1394   43

if (subst==TRUE) uvDB<-uvDB[seq(1,nrow(uvDB),by=5),]
dim(uvDB)

## [1] 1394   43

## use StopGain vs Synonymous for Training lables:
newTrainingLabel<-read.csv("variant_type_for_train.csv")
kdel<-unique(newTrainingLabel$uPOS[newTrainingLabel$EventType=="StopGain"])
kneut<-unique(newTrainingLabel$uPOS[newTrainingLabel$EventType=="Synonymous"])
uvDB$label<-rep(NA,nrow(uvDB))
uvDB$label[uvDB$uPOS %in% kdel]<-"P"
uvDB$label[uvDB$uPOS %in% kneut]<-"B"
head(uvDB)

##   exon         uPOS X.CHROM      POS REF ALT AApos AltAA AltCodon         EventType RefAA
## 1   15 32356417_T_A   chr13 32356417   T   A    NA  <NA>     <NA> UpstreamNoncoding  <NA>
## 2   15 32356418_C_T   chr13 32356418   C   T    NA  <NA>     <NA> UpstreamNoncoding  <NA>
## 3   15 32356420_T_C   chr13 32356420   T   C    NA  <NA>     <NA> UpstreamNoncoding  <NA>
## 4   15 32356422_G_A   chr13 32356422   G   A    NA  <NA>     <NA> UpstreamNoncoding  <NA>
## 5   15 32356423_A_T   chr13 32356423   A   T    NA  <NA>     <NA> UpstreamNoncoding  <NA>
## 6   15 32356425_A_G   chr13 32356425   A   G    NA  <NA>     <NA> UpstreamNoncoding  <NA>
##   RefCodon SpliceAI_ALLELE SpliceAI_DP_AG SpliceAI_DP_AL SpliceAI_DP_DG SpliceAI_DP_DL
## 1     <NA>               A            -7            10           -45            10
## 2     <NA>               T             9            -8            38             9
## 3     <NA>               C            27             7           -48             7
## 4     <NA>               A             5           -12            34             5
## 5     <NA>               T             4           -13             4           -13
## 6     <NA>               G            15             2           -17             2
##   SpliceAI_DS_AG SpliceAI_DS_AL SpliceAI_DS_DG SpliceAI_DS_DL SpliceAI_GENE AA_length R1_lib
## 1           0.00           0.28              0              0         BRCA2         0    225
## 2           0.01           0.00              0              0         BRCA2         0    209
```

```
## 3           0.02           0.03              0              0         BRCA2         0    169
## 4           0.01           0.00              0              0         BRCA2         0    134
## 5           0.01           0.01              0              0         BRCA2         0    379
## 6           0.54           0.95              0              0         BRCA2         0    685
##   R1_D5 R1_D14 R2_lib R2_D5 R2_D14 R3_lib R3_D5 R3_D14 R4_lib R4_D5 R4_D14 R5_lib R5_D5 R5_D14
## 1   369    279    299   225     78   4063  9016   5502     NA    NA     NA     NA    NA     NA
## 2   299    130    285   181    122   2785  4675   2924     NA    NA     NA     NA    NA     NA
## 3   359    234    244   293    115   2109  3828   2971     NA    NA     NA     NA    NA     NA
## 4   101     78    147    86     55   1664  1504   1413     NA    NA     NA     NA    NA     NA
## 5   835    410    455   564    243   5015  8691   8999     NA    NA     NA     NA    NA     NA
## 6   964    242    912   904    461  10200 13180   8213     NA    NA     NA     NA    NA     NA
##   R6_lib R6_D5 R6_D14 class   PB label
## 1     NA    NA     NA  <NA> <NA>  <NA>
## 2     NA    NA     NA  <NA> <NA>  <NA>
## 3     NA    NA     NA  <NA> <NA>  <NA>
## 4     NA    NA     NA  <NA> <NA>  <NA>
## 5     NA    NA     NA  <NA> <NA>  <NA>
## 6     NA    NA     NA  <NA> <NA>  <NA>
```

4 DB Row Column Names

Add row names; change a few column names; make 'Exon' a factor:

```
if ((db=="uvCounts")|(db=="uvCountsFnl")){
   colnames(uvDB)[colnames(uvDB)=="class"]<-"classification"
   colnames(uvDB)[colnames(uvDB)=="exon"]<-"Exon"
   colnames(uvDB)[colnames(uvDB)=="PB"]<-"P_B"
   ## Annotation DB:
   annot<-uvDB[,c(1:23,42:43)]
}
if (nrow(uvDB)==length(unique(uvDB$uPOS))) rownames(uvDB)<-uvDB$uPOS
uvDB$Exon<-factor(uvDB$Exon)
```

4.1 Compute Offsets and Other Summaries for Counts Data

Compute count totals for all observed combinations of day, exon and replicate (labeled 'offsets' in the code below), then use these to compute raw abundance rates (denoted $A_{v,r,d}$ in the Supplement). Use the abundance rates to compute day 14 to day 5 and day 14 to day 0 (lib) rate ratios (denoted $R_{v,r}^{14:5}$ and $R_{v,r}^{14:0}$, respectively, in the Supplement). Add columns containing the 'offsets' and the two rate ratios to the working data base `uvDB`. Finally, compute an exon–specific standardized position variable `PosStd` and add it to the working data base.

```
if ((db=="uvCounts")|(db=="uvCountsFnl")){
  cnames<-colnames(uvDB)
  lcnames<-nchar(cnames)
  daylib<-cnames[substr(cnames,lcnames-2,lcnames)=="lib"]
  day5<-cnames[substr(cnames,lcnames-1,lcnames)=="D5"]
  day14<-cnames[substr(cnames,lcnames-2,lcnames)=="D14"]
  offsetlib<-matrix(NA,nrow(uvDB),length(daylib))
```

```
  colnames(offsetlib)<-cnOffsetLib<-paste0("R",1:6,"offsetLib")
  offset5<-matrix(NA,nrow(uvDB),length(day5))
  colnames(offset5)<-cnOffset5<-paste0("R",1:6,"offsetD5")
  offset14<-matrix(NA,nrow(uvDB),length(day14))
  colnames(offset14)<-cnOffset14<-paste0("R",1:6,"offsetD14")
  for (i in 1:length(day5)){
      templib<-lapply(split(uvDB[,daylib[i]],uvDB$Exon),sum,na.rm=TRUE)
      temp5<-lapply(split(uvDB[,day5[i]],uvDB$Exon),sum,na.rm=TRUE)
      temp14<-lapply(split(uvDB[,day14[i]],uvDB$Exon),sum,na.rm=TRUE)
      offsetlib[,i]<-as.numeric(templib[uvDB$Exon])
      offset5[,i]<-as.numeric(temp5[uvDB$Exon])
      offset14[,i]<-as.numeric(temp14[uvDB$Exon])
  }
  rawlib<-(uvDB[,daylib]/offsetlib)  ## lib (day 0) abundance rates
  raw5<-(uvDB[,day5]/offset5)        ## day 5 abundance rates
  raw14<-(uvDB[,day14]/offset14)     ## day 14 abundance rates
  raw<-(raw14/raw5)                  ## day 14 to day 5 rate ratio
  raw2<-(raw14/rawlib)               ## day 15 to day 0 rate ratio
  colnames(raw)<-paste0("R",1:6,"_raw")
  colnames(raw2)<-paste0("R",1:6,"_rawlib")
  uvDB<-cbind(uvDB,offsetlib,offset5,offset14,raw,raw2)
  rm(offsetlib,offset5,offset14,lcnames,raw,raw2,rawlib,raw5,raw14)
  pos.min<-lapply(split(uvDB[,"POS"],uvDB$Exon),min,na.rm=TRUE)
  pos.max<-lapply(split(uvDB[,"POS"],uvDB$Exon),max,na.rm=TRUE)
  pos.range<- as.numeric(pos.max) - as.numeric(pos.min)
  names(pos.range)<-names(pos.min)
  uvDB$PosMin<-as.numeric(pos.min[uvDB$Exon])
  uvDB$PosMax<-as.numeric(pos.max[uvDB$Exon])
  uvDB$PosRange<-as.numeric(pos.range[uvDB$Exon])
  uvDB$PosStd<-((uvDB$POS - uvDB$PosMin)/(uvDB$PosMax - uvDB$PosMin))
  summary(uvDB$PosStd)
}
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2513  0.5000  0.5009  0.7532  1.0000
```

5 Exploratory Data Analysis

5.1 Tabular Summaries

Cross–tabulations of several design and annotation variables:

```
table(uvDB$Exon,uvDB$EventType,useNA="always")
```

```
##
##        DownstreamNonCoding DownstreamNonCodingAndPartialCodingDownstream Missense
##   15                     6                                             0       76
##   16                     7                                             0       78
##   17                     6                                             0       76
##   18C                    5                                             1       74
```

```
##   18N                    0                                          0       70
##   19                     6                                          0       64
##   20                     6                                          0       60
##   21                     6                                          0       55
##   22                     5                                          1       90
##   23                     5                                          1       66
##   24                     6                                          0       58
##   25C                    6                                          0       49
##   25N                    0                                          0       51
##   26                     0                                          0       31
##   <NA>                   0                                          0        0
##
##        PartialCodingDownStream PartialCodingUpStream StopGain Synonymous UpstreamNoncoding <NA>
##   15                         0                      2        4         26                 6    0
##   16                         1                      0       11         21                 6    0
##   17                         2                      0        1         22                 6    0
##   18C                        0                      0        8         24                 0    0
##   18N                        0                      1        6         28                 6    0
##   19                         0                      0        6         22                 6    0
##   20                         1                      0        9         16                 6    0
##   21                         0                      1        4         13                 6    0
##   22                         0                      0        8         19                 6    0
##   23                         0                      2        7         22                 6    0
##   24                         1                      0        6         17                 6    0
##   25C                        0                      0        5         18                 0    0
##   25N                        0                      1        4         15                 6    0
##   26                         0                      0        0          3                 6    0
##   <NA>                       0                      0        0          0                 0    0
```

```
table(uvDB$Exon,uvDB$P_B,useNA="always")
```

```
##
##          B   P <NA>
##   15     5   2  113
##   16     2   6  116
##   17     2   5  106
##   18C    5   7  100
##   18N   11   4   96
##   19     5   4   95
##   20     3   4   91
##   21     4   3   78
##   22     6   5  118
##   23     2   5  102
##   24     2   4   88
##   25C    6   2   70
##   25N    4   5   68
##   26     2   0   38
##   <NA>   0   0    0
```

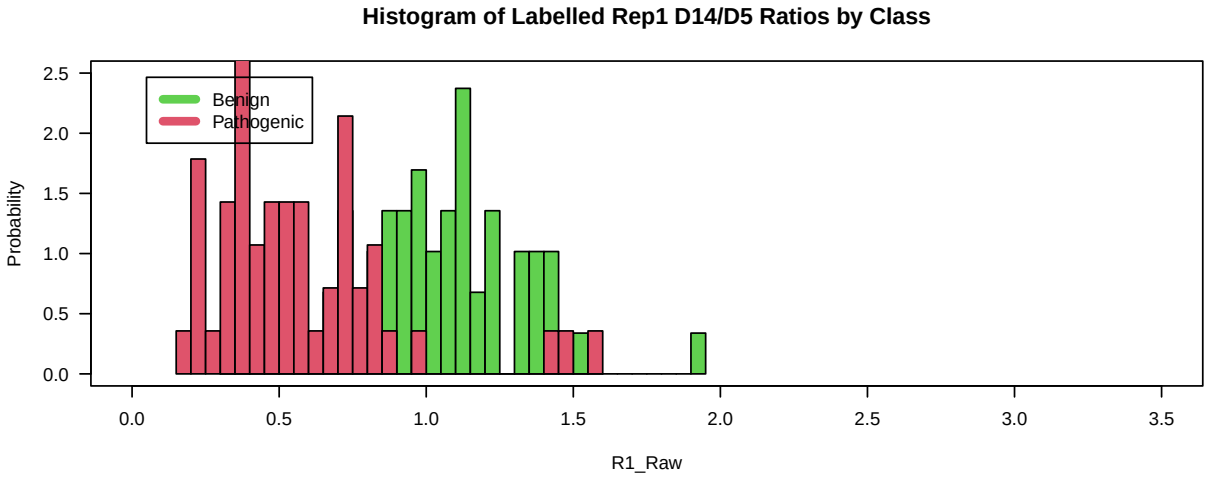
```
##table(uvDB£EventType,uvDB£spliceR,useNA="always")
table(uvDB$P_B,uvDB$classification)
```

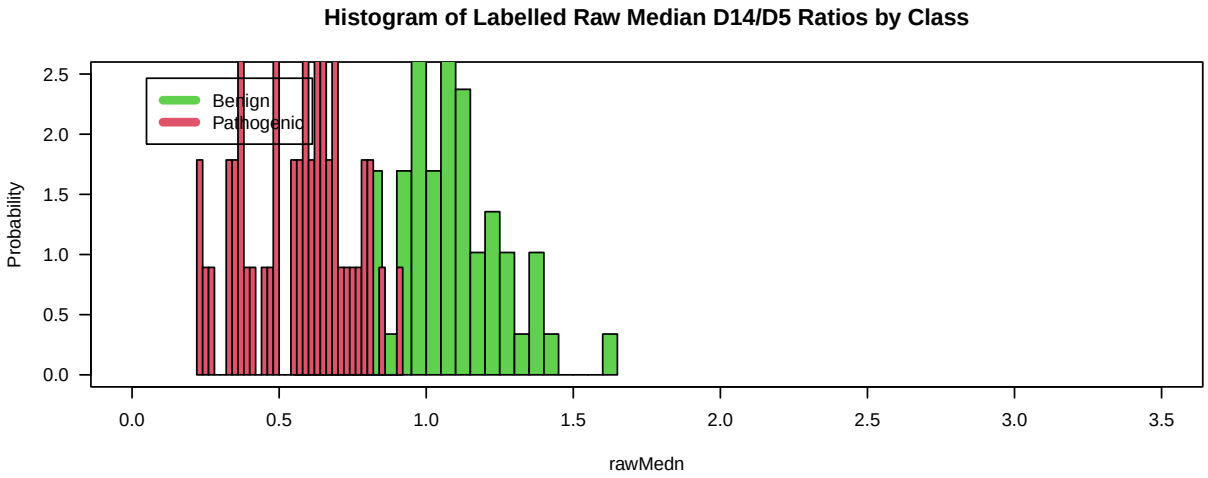
```
##
##      B B/LB LB LP  P P/LP
##   B  8    2 49  0  0    0
##   P  0    0  0  2 48    6

table(uvDB$label,uvDB$classification)

##
##      B B/LB LB LP  P P/LP
##   B  1    0 49  0  0    0
##   P  0    0  0  0 39    1

##table(uvDB£EventType,uvDB£spliceR)
table(uvDB$EventType,uvDB$classification)

##
##                                                B B/LB LB LP  P P/LP
##   DownstreamNonCoding                          0    0  0  0  0    0
##   DownstreamNonCodingAndPartialCodingDownstream 0    0  0  0  0    0
##   Missense                                     7    2  0  2  4    5
##   PartialCodingDownStream                      0    0  0  0  1    0
##   PartialCodingUpStream                        0    0  0  0  0    0
##   StopGain                                     0    0  0  0 43    1
##   Synonymous                                   1    0 49  0  0    0
##   UpstreamNoncoding                            0    0  0  0  0    0
```

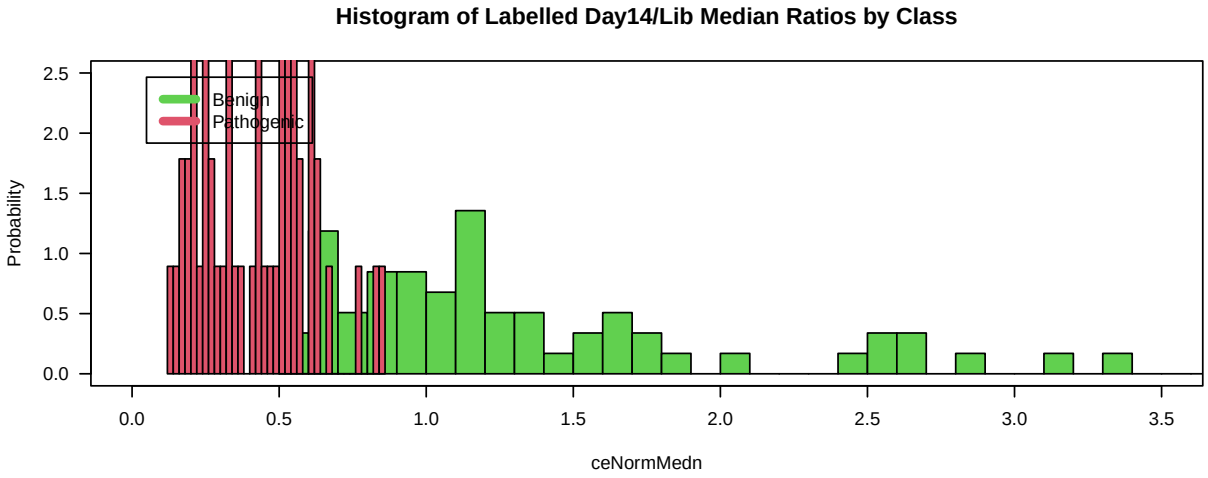
5.2 Graphical Summaries of Labeled Data

Histograms of: (1) the replicate 1 day 14 to day 5 rate ratio, (2) the median day 14 to day 5 rate ratio across replicates and (3) the median day 14 to day 0 (not yet positionally normalized) rate ratio.

```
par(mfrow=c(3,1))
plot(c(0,3.5),c(0,2.5),las=1,xlab="R1_Raw",ylab="Probability",type="n",
    ,main="Histogram of Labelled Rep1 D14/D5 Ratios by Class")
hist(uvDB$R1_raw[(!is.na(uvDB$P_B))&(uvDB$P_B=="B")],
     col=3,prob=TRUE,nclass=30,add=TRUE)
hist(uvDB$R1_raw[(!is.na(uvDB$P_B))&(uvDB$P_B=="P")],
     col=2,prob=TRUE,nclass=30,add=TRUE)
legend("topleft",inset=0.05,legend=c("Benign","Pathogenic"),
       col=c(3,2),lwd=5)
## Median of Raw D14/D5 Ratios:
uvDB$rawMedn<-apply(uvDB[,c("R1_raw","R2_raw","R3_raw","R4_raw","R5_raw","R6_raw")],
                    1,FUN=median,na.rm=TRUE)
plot(c(0,3.5),c(0,2.5),las=1,xlab="rawMedn",ylab="Probability",type="n",
    ,main="Histogram of Labelled Raw Median D14/D5 Ratios by Class")
hist(uvDB$rawMedn[(!is.na(uvDB$P_B))&(uvDB$P_B=="B")],
     col=3,prob=TRUE,nclass=30,add=TRUE)
hist(uvDB$rawMedn[(!is.na(uvDB$P_B))&(uvDB$P_B=="P")],
     col=2,prob=TRUE,nclass=30,add=TRUE)
legend("topleft",inset=0.05,legend=c("Benign","Pathogenic"),
       col=c(3,2),lwd=5)
## Median of Raw Day14/lib Ratios
uvDB$rawLibMedn<-apply(uvDB[,c("R1_rawlib","R2_rawlib","R3_rawlib","R4_rawlib",
                               "R5_rawlib","R6_rawlib")],
                       1,FUN=median,na.rm=TRUE)
plot(c(0,3.5),c(0,2.5),las=1,xlab="ceNormMedn",ylab="Probability",type="n",
    ,main="Histogram of Labelled Day14/Lib Median Ratios by Class")
hist(uvDB$rawLibMedn[(!is.na(uvDB$P_B))&(uvDB$P_B=="B")],
     col=3,prob=TRUE,nclass=30,add=TRUE)
hist(uvDB$rawLibMedn[(!is.na(uvDB$P_B))&(uvDB$P_B=="P")],
     col=2,prob=TRUE,nclass=30,add=TRUE)
legend("topleft",inset=0.05,legend=c("Benign","Pathogenic"),
       col=c(3,2),lwd=5)
```


Histogram of Labelled Rep1 D14/D5 Ratios by Class
Benign
Pathogenic
Probability
R1_Raw


Histogram of Labelled Raw Median D14/D5 Ratios by Class
Benign
Pathogenic
Probability
rawMedn


Histogram of Labelled Day14/Lib Median Ratios by Class
Benign
Pathogenic
Probability
ceNormMedn

6 Create 'Tall' Data Structure

Here we create a stacked data structure with day– and replicate–specific measurments in different rows and the variant counts and batch totals ('offsets') each in their own column. This format is needed modeling. Save the old `uvDB` as `uvMaster` and save the stacked data structure as the new `uvDB`. Convert categorical variables from character to factor types.

```
uvMaster<-uvDB
uvDB<-uvDB[,c("uPOS","PosStd","Exon","P_B","label","EventType",day5,day14,daylib,cnOffset5,cnOffset14,cn
rownames(uvDB)<-NULL
temp<-uvDB
n<-nrow(temp)
tall<-NULL
for (i in 1:6){
  replic<-paste0("R",i)
  d5<-cbind(as.matrix(temp[,c("uPOS","PosStd","Exon","P_B","label","EventType",day5[i],cnOffset5[i])]),
            rep("D5",n),rep(replic,n))
  d14<-cbind(as.matrix(temp[,c("uPOS","PosStd","Exon","P_B","label","EventType",day14[i],cnOffset14[i])]
            rep("D14",n),rep(replic,n))
  d0<-cbind(as.matrix(temp[,c("uPOS","PosStd","Exon","P_B","label","EventType",daylib[i],cnOffsetLib[i]]
            rep("D0",n),rep(replic,n))
  colnames(d0)<-colnames(d5)<-colnames(d14)<-NULL
  tall<-rbind(tall,d0,d5,d14)
}
uvDB<-data.frame(tall)
rm(temp)
colnames(uvDB)<-c("variant","PosStd","Exon","p.b","label","eventtype","count","offset","day","replicate"
uvDB$PosStd<-as.numeric(uvDB$PosStd)
uvDB$count<-as.numeric(uvDB$count)
uvDB$offset<-as.numeric(uvDB$offset)
uvDB<-uvDB[!is.na(uvDB$count),]
uvDB$day<-factor(uvDB$day)
uvDB$replicate<-factor(uvDB$replicate)
uvDB$variant<-factor(uvDB$variant)
uvDB$Exon<-factor(uvDB$Exon)
uvDB$p.b<-factor(uvDB$p.b)
uvDB$p.b<-factor(uvDB$label)
uvDB$batchE<-as.numeric(uvDB$Exon)   ## batch=exon
## Exon by Rep
uvDB$ER<-paste0(substr(uvDB$Exon,1,4),uvDB$replicate)
uvDB$ER<-factor(uvDB$ER)
uvDB$batch<-as.numeric(uvDB$ER)  ## More refined batch variable
table(uvDB$ER)
```

```
##
##  15R1  15R2  15R3  16R1  16R2  16R3  17R1  17R2  17R3 18CR1 18CR2 18CR3 18CR4 18CR5 18NR1 18NR2
##   360   360   360   372   372   372   339   339   339   330   330   330   330   330   333   330
## 18NR3 18NR4  19R1  19R2  19R3  20R1  20R2  20R3  21R1  21R2  21R3  21R4  21R5  21R6  22R1  22R2
##   333   327   312   312   312   294   294   294   249   249   249   249   249   249   387   387
##  22R3  23R1  23R2  23R3  24R1  24R2  24R3 25CR1 25CR2 25CR3 25NR1 25NR2 25NR3  26R1  26R2  26R3
##   387   327   327   327   282   282   282   234   234   234   231   231   231   120   120   120
```

```
table(uvDB$batch)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## 360 360 360 372 372 372 339 339 339 330 330 330 330 330 333 330 333 327 312 312 312 294 294 294
##  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48
## 249 249 249 249 249 249 387 387 387 327 327 327 282 282 282 234 234 234 231 231 231 120 120 120
```

```
head(uvDB)
```

```
##        variant      PosStd Exon  p.b label         eventtype count offset day replicate batchE
## 1 32356417_T_A 0.000000000   15 <NA>  <NA> UpstreamNoncoding   225  33900  D0        R1       1
## 2 32356418_C_T 0.004975124   15 <NA>  <NA> UpstreamNoncoding   209  33900  D0        R1       1
## 3 32356420_T_C 0.014925373   15 <NA>  <NA> UpstreamNoncoding   169  33900  D0        R1       1
## 4 32356422_G_A 0.024875622   15 <NA>  <NA> UpstreamNoncoding   134  33900  D0        R1       1
## 5 32356423_A_T 0.029850746   15 <NA>  <NA> UpstreamNoncoding   379  33900  D0        R1       1
## 6 32356425_A_G 0.039800995   15 <NA>  <NA> UpstreamNoncoding   685  33900  D0        R1       1
##     ER batch
## 1 15R1     1
## 2 15R1     1
## 3 15R1     1
## 4 15R1     1
## 5 15R1     1
## 6 15R1     1
```

```
summary(uvDB)
```

```
##          variant          PosStd            Exon        p.b           label
##  32376660_T_C:   18   Min.   :0.0000   18C    :1650   B   : 2619   Length:14241
##  32376662_T_A:   18   1st Qu.:0.2500   21     :1494   P   :  738   Class :character
##  32376663_T_G:   18   Median :0.5000   18N    :1323   NA's:10884   Mode  :character
##  32376665_T_C:   18   Mean   :0.5008   22     :1161
##  32376667_A_C:   18   3rd Qu.:0.7538   16     :1116
##  32376668_G_T:   18   Max.   :1.0000   15     :1080
##  (Other)     :14133                    (Other):6417
##   eventtype             count            offset        day       replicate     batchE
##  Length:14241       Min.   :     4   Min.   :  7984   D0 :4747   R1:4170   Min.   : 1.000
##  Class :character   1st Qu.:  1130   1st Qu.:305404   D14:4747   R2:4167   1st Qu.: 4.000
##  Mode  :character   Median :  2828   Median :441122   D5 :4747   R3:4170   Median : 6.000
##                     Mean   :  3980   Mean   :396404              R4: 906   Mean   : 6.696
##                     3rd Qu.:  4988   3rd Qu.:530448              R5: 579   3rd Qu.:10.000
##                     Max.   :193125   Max.   :926803              R6: 249   Max.   :14.000
##
##        ER            batch
##  22R1   :  387   Min.   : 1.00
##  22R2   :  387   1st Qu.:11.00
##  22R3   :  387   Median :21.00
##  16R1   :  372   Mean   :22.29
##  16R2   :  372   3rd Qu.:34.00
##  16R3   :  372   Max.   :48.00
##  (Other):11964
```

7 Positional Normalization

Computations in support of the positional normalization of the day 0 (lib) counts to the day 5 values. Two methods for computing the positional correction are implemented here: by exon and by exon and replicate. We use the latter approach; the former is not used.

7.0.1 Data Processing

Create a data structure (`uvNorm`) for the normalization model. Compute the day 5 to day 0 rate ratio $R^{5:0}$ for the model and subset to variants such that $R^{5:0} > 0.5$ or $R^{14:5} > 0.8$.

```
tall<-data.frame(tall)
colnames(tall)<-c("variant","PosStd","Exon","p.b","label","eventtype","count","offset","day","replicate
## Exon by Rep
tall$ER<-paste0(substr(tall$Exon,1,4),tall$replicate)
table(tall$ER)

##
##  15R1  15R2  15R3  15R4  15R5  15R6  16R1  16R2  16R3  16R4  16R5  16R6  17R1  17R2  17R3  17R4
##   360   360   360   360   360   360   372   372   372   372   372   372   339   339   339   339
##  17R5  17R6 18CR1 18CR2 18CR3 18CR4 18CR5 18CR6 18NR1 18NR2 18NR3 18NR4 18NR5 18NR6  19R1  19R2
##   339   339   336   336   336   336   336   336   333   333   333   333   333   333   312   312
##  19R3  19R4  19R5  19R6  20R1  20R2  20R3  20R4  20R5  20R6  21R1  21R2  21R3  21R4  21R5  21R6
##   312   312   312   312   294   294   294   294   294   294   255   255   255   255   255   255
##  22R1  22R2  22R3  22R4  22R5  22R6  23R1  23R2  23R3  23R4  23R5  23R6  24R1  24R2  24R3  24R4
##   387   387   387   387   387   387   327   327   327   327   327   327   282   282   282   282
##  24R5  24R6 25CR1 25CR2 25CR3 25CR4 25CR5 25CR6 25NR1 25NR2 25NR3 25NR4 25NR5 25NR6  26R1  26R2
##   282   282   234   234   234   234   234   234   231   231   231   231   231   231   120   120
##  26R3  26R4  26R5  26R6
##   120   120   120   120

table(tall$eventtype)

##
##                            DownstreamNonCoding DownstreamNonCodingAndPartialCodingDownstream
##                                           1152                                           54
##                                       Missense                       PartialCodingDownStream
##                                          16164                                           90
##                              PartialCodingUpStream                                   StopGain
##                                            126                                         1422
##                                     Synonymous                             UpstreamNoncoding
##                                           4788                                         1296

table(tall$day)

##
##   D0  D14   D5
## 8364 8364 8364

uvD0<-tall[tall$day=="D0",]
uvD5<-tall[tall$day=="D5",]
uvD14<-tall[tall$day=="D14",]
table((uvD0$variant==uvD5$variant)&(uvD0$Exon==uvD5$Exon)&(uvD0$replicate==uvD5$replicate))
```

```
##
## TRUE
## 8364

table((uvD14$variant==uvD5$variant)&(uvD14$Exon==uvD5$Exon)&(uvD14$replicate==uvD5$replicate))

##
## TRUE
## 8364

uvD0$rawR<-(as.numeric(uvD0$count)/as.numeric(uvD0$offset))
uvD5$rawR<-(as.numeric(uvD5$count)/as.numeric(uvD5$offset))
uvD14$rawR<-(as.numeric(uvD14$count)/as.numeric(uvD14$offset))
uvD5$R14.5<-(uvD14$rawR/uvD5$rawR)
uvD5$R5.0<-(uvD5$rawR/uvD0$rawR)
R14.5<-lapply(split(uvD5[,"R14.5"],uvD5$variant),mean,na.rm=TRUE)
R14.5<-unlist(R14.5)
R5.0<-lapply(split(uvD5[,"R5.0"],uvD5$variant),mean,na.rm=TRUE)
R5.0<-unlist(R5.0)
table(names(R14.5)==names(R5.0))

##
## TRUE
## 1394

## Subset of Variants to Use for Positional Normalization of Lib Values
posTrainVariants<-names(R5.0)[(R14.5>0.8)|(R5.0>0.5)]
length(posTrainVariants)

## [1] 1359

keep<-(uvD5$variant %in% posTrainVariants)
## Ratio for Positional Normalization Analysis:
uvD5$Ratio<-(as.numeric(uvD5$count)/as.numeric(uvD5$offset))
uvD5$Ratio<-uvD5$Ratio/(as.numeric(uvD0$count)/as.numeric(uvD0$offset))
summary(uvD5$Ratio)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.063   0.726   0.990   1.045   1.207   8.625    3617

##keep<-((uvD5$eventtype=="Synonymous")|(uvD5$eventtype=="Missense"))
keep<-(keep&(!is.na(uvD5$Ratio)))
uvNorm<-uvD5[keep,]
uvNorm$Exon<-as.factor(uvNorm$Exon)
uvNorm$PosStd<-as.numeric(uvNorm$PosStd)
uvNorm$lRatio<-log(uvNorm$Ratio)
```

7.0.2 Normalization By Exon, Combining Replicates (Not Used)

Implement the exon–by–exon normalization scheme and generate plots of the data and the estimated positional normalization curve.

```
gam.pos<-mgcv::gam(lRatio ~  s(PosStd,bs="ad",by=Exon),data=uvNorm)
gam.pos2<-mgcv::gam(lRatio ~ s(PosStd,bs="tp",by=Exon),data=uvNorm)
temp<-predict(gam.pos,newdata=uvNorm,se.fit=TRUE)
uvNorm$adSmooth<-temp$fit
uvNorm$adSmoothSE<-temp$se.fit
temp<-predict(gam.pos2,newdata=uvNorm,se.fit=TRUE)
uvNorm$tpSmooth<-temp$fit
uvNorm$tpSmoothSE<-temp$se.fit
posFits<-unique(uvNorm[,c("variant","Exon","PosStd","adSmooth","adSmoothSE",
                          "tpSmooth","tpSmoothSE")])
uExon<-unique(as.character(uvNorm$Exon))
nExon<-length(uExon)
pdf("SmoothFitsFnl.pdf",height=8.5,width=11)
par(mfrow=c(2,1))
## Adaptive Smoooth
for (i in 1:nExon){
plot(uvNorm$PosStd[uvNorm$Exon==uExon[i]],
     uvNorm$lRatio[uvNorm$Exon==uExon[i]],las=1,
     ylab="logRatio",xlab="Standardized Position",
     main=paste0("Adaptive Smooth Fit to Exon ",uExon[i]),xlim=c(0,1.12))
uRep<-unique(uvNorm$replicate[(uvNorm$Exon==uExon[i])])
for (j in 1:length(uRep)){
  points(uvNorm$PosStd[(uvNorm$Exon==uExon[i])&(uvNorm$replicate==uRep[j])],
       uvNorm$lRatio[(uvNorm$Exon==uExon[i])&(uvNorm$replicate==uRep[j])],
       col=j,pch=16)
}
##points(uvNorm£PosStd[(uvNorm£Exon==uExon[i])&(uvNorm£eventtype=="Synonymous")],
##     uvNorm£lRatio[(uvNorm£Exon==uExon[i])&(uvNorm£eventtype=="Synonymous")],
##     col=3,pch=16)
legend("topright",inset=0.01,col=c(1:length(uRep)),lwd=5,legend=uRep)
##legend("topright",inset=0.05,col=c(1,3),lwd=5,legend=c("Missense","Synonymous"))
lines(posFits$PosStd[posFits$Exon==uExon[i]],
      posFits$adSmooth[posFits$Exon==uExon[i]],col=2,lwd=3)
lines(posFits$PosStd[posFits$Exon==uExon[i]],
      posFits$adSmooth[posFits$Exon==uExon[i]]+2*posFits$adSmoothSE[posFits$Exon==uExon[i]],
      col=2,lwd=3,lty=2)
lines(posFits$PosStd[posFits$Exon==uExon[i]],
      posFits$adSmooth[posFits$Exon==uExon[i]]-2*posFits$adSmoothSE[posFits$Exon==uExon[i]],
      col=2,lwd=3,lty=2)
## Thin Plate Smoooth
plot(uvNorm$PosStd[uvNorm$Exon==uExon[i]],
     uvNorm$lRatio[uvNorm$Exon==uExon[i]],las=1,
     ylab="logRatio",xlab="Standardized Position",
     main=paste0("Thin Plate Smooth Fit to Exon ",uExon[i]),
     type="n",xlim=c(0,1.12))
for (j in 1:length(uRep)){
  points(uvNorm$PosStd[(uvNorm$Exon==uExon[i])&(uvNorm$replicate==uRep[j])],
       uvNorm$lRatio[(uvNorm$Exon==uExon[i])&(uvNorm$replicate==uRep[j])],
       col=j,pch=16)
}
##points(uvNorm£PosStd[(uvNorm£Exon==uExon[i])&(uvNorm£eventtype=="Synonymous")],
```

```
##     uvNorm$lRatio[(uvNorm$Exon==uExon[i])&(uvNorm$eventtype=="Synonymous")],
##     col=3,pch=16)
legend("topright",inset=0.01,col=c(1:length(uRep)),lwd=5,legend=uRep)
##legend("topright",inset=0.05,col=c(1,3),lwd=5,legend=c("Missense","Synonymous"))
lines(posFits$PosStd[posFits$Exon==uExon[i]],
      posFits$tpSmooth[posFits$Exon==uExon[i]],col=2,lwd=3)
lines(posFits$PosStd[posFits$Exon==uExon[i]],
      posFits$tpSmooth[posFits$Exon==uExon[i]]+2*posFits$tpSmoothSE[posFits$Exon==uExon[i]],
      col=2,lwd=3,lty=2)
lines(posFits$PosStd[posFits$Exon==uExon[i]],
      posFits$tpSmooth[posFits$Exon==uExon[i]]-2*posFits$tpSmoothSE[posFits$Exon==uExon[i]],
      col=2,lwd=3,lty=2)
}
dev.off()

## pdf
##   2
```

7.0.3 Normalization By Exon and Replicate (This Method Used)

Implement the exon by replicate normalization scheme and generate plots of the data and the estimated positional normalization curve.

```
uvNorm$adSmooth<-rep(NA,nrow(uvNorm))
uvNorm$adSmoothSE<-rep(NA,nrow(uvNorm))
uvDB$adSmooth<-rep(NA,nrow(uvDB))
uvDB$adSmoothSE<-rep(NA,nrow(uvDB))
pdf("SmoothFitsByRepFnl.pdf",height=8.0,width=24)
par(mfrow=c(1,1))
## Adaptive Smoooth -- Independent Fits by Exon and Replicate
for (i in 1:nExon){
    plot(uvNorm$PosStd[uvNorm$Exon==uExon[i]],
         uvNorm$lRatio[uvNorm$Exon==uExon[i]],las=1,
         ylab="logRatio",xlab="Standardized Position",
         main=paste0("Replicate-Specific Adaptive Smooth Fit to Exon ",uExon[i]),
         xlim=c(0,1.12))
    uRep<-unique(uvNorm$replicate[(uvNorm$Exon==uExon[i])])  ##unique replicate IDs
    for (j in 1:length(uRep)){
        keep<-((uvNorm$Exon==uExon[i])&(uvNorm$replicate==uRep[j]))
        gam.pos<-mgcv::gam(lRatio ~  s(PosStd,bs="ad",by=Exon),data=uvNorm[keep,])
        temp<-predict(gam.pos,newdata=uvNorm[keep,],se.fit=TRUE)
        uvNorm$adSmooth[keep]<-temp$fit
        uvNorm$adSmoothSE[keep]<-temp$se.fit
        points(uvNorm$PosStd[keep],uvNorm$lRatio[keep],col=j,pch=16)
        posFits<-unique(uvNorm[keep,c("variant","Exon","PosStd","adSmooth","adSmoothSE")])
        lines(posFits$PosStd,posFits$adSmooth,col=j,lwd=5)
        ##lines(posFits$PosStd,posFits$adSmooth + 2*posFits$adSmoothSE,
        ##      col=j,lwd=3,lty=2)
        ##lines(posFits$PosStd,posFits$adSmooth - 2*posFits$adSmoothSE,
        ##      col=j,lwd=3,lty=2)
        ## Add Effects Predictions to Full DB:
```

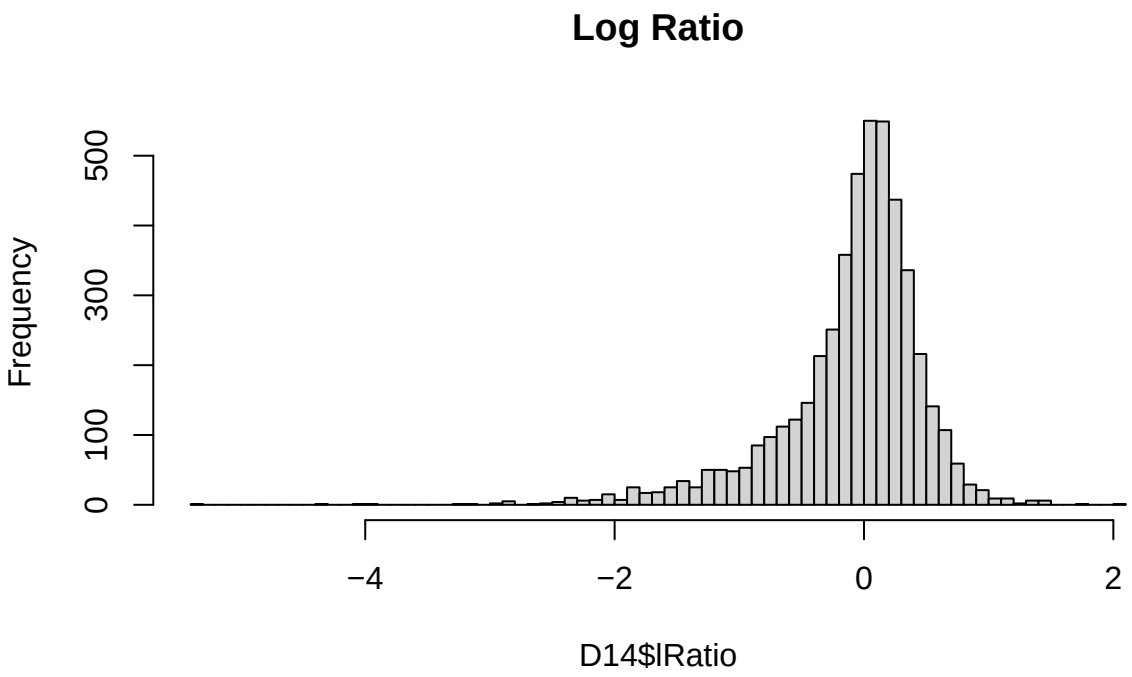
```
        keep2<-((uvDB$Exon==uExon[i])&(uvDB$replicate==uRep[j]))
        temp2<-predict(gam.pos,newdata=uvDB[keep2,],se.fit=TRUE)
        uvDB$adSmooth[keep2]<-temp2$fit
        uvDB$adSmoothSE[keep2]<-temp2$se.fit
    }
    legend("topright",inset=0.01,col=c(1:length(uRep)),lwd=5,legend=uRep)

}
dev.off()

## pdf
##   2
```

8 Exploratory Mixed Effects Model of The Labeled Data

8.1 Setup Data Structure

The following code chunk creates the final analysis data set and adds the positionally normalized, logged day 14 to day 0 rate ratio and the exon by replicate batch variable. This structure becomes the new data base matrix `uvDB`. This structure is used for this section's exploratory analysis and for the final VarCall analysis.

```
## Positional Normalization for Day 0:
uvDB$offset[uvDB$day=="D0"]<-(uvDB$offset[uvDB$day=="D0"]/exp(uvDB$adSmooth[uvDB$day=="D0"]))
D0<-uvDB[uvDB$day=="D0",]
D0$ID<-paste0(D0$variant,"_",D0$replicate)
length(unique(D0$ID))

## [1] 4747

D14<-uvDB[uvDB$day=="D14",]
D14$ID<-paste0(D14$variant,"_",D14$replicate)
length(unique(D14$ID))

## [1] 4747

table(D14$ID %in% D0$ID)

##
## TRUE
## 4747

table(D0$ID %in% D14$ID)

##
## TRUE
## 4747

D0<-D0[D0$ID %in% D14$ID,]
table(D0$ID == D14$ID)

##
## TRUE
## 4747

##############################################################################
## This is the Normalized log(Ratio) Used in the Classification Model: ##
##############################################################################
D14$lRatio<-(log(D14$count) - log(D14$offset)) - (log(D0$count) - log(D0$offset))
hist(D14$lRatio,nclass=100,main="Log Ratio")
```




```
##########################################
## Data Set for Analysis of Labeled Data: ##
##########################################
uvKnown<-D14[!is.na(D14$label),]
#################################
## Data Set for Full Analysis: ##
#################################
uvDB<-D14
uvDB$ER<-factor(uvDB$ER)
uvDB$batch<-as.numeric(uvDB$ER)  ## More refined batch variable
table(uvDB$ER)

##
##  15R1  15R2  15R3  16R1  16R2  16R3  17R1  17R2  17R3 18CR1 18CR2 18CR3 18CR4 18CR5 18NR1 18NR2
##   120   120   120   124   124   124   113   113   113   110   110   110   110   110   111   110
## 18NR3 18NR4  19R1  19R2  19R3  20R1  20R2  20R3  21R1  21R2  21R3  21R4  21R5  21R6  22R1  22R2
##   111   109   104   104   104    98    98    98    83    83    83    83    83    83   129   129
##  22R3  23R1  23R2  23R3  24R1  24R2  24R3 25CR1 25CR2 25CR3 25NR1 25NR2 25NR3  26R1  26R2  26R3
##   129   109   109   109    94    94    94    78    78    78    77    77    77    40    40    40

table(uvDB$batch)

##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## 120 120 120 124 124 124 113 113 113 110 110 110 110 110 111 110 111 109 104 104 104  98  98  98
##  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48
##  83  83  83  83  83  83 129 129 129 109 109 109  94  94  94  78  78  78  77  77  77  40  40  40
```

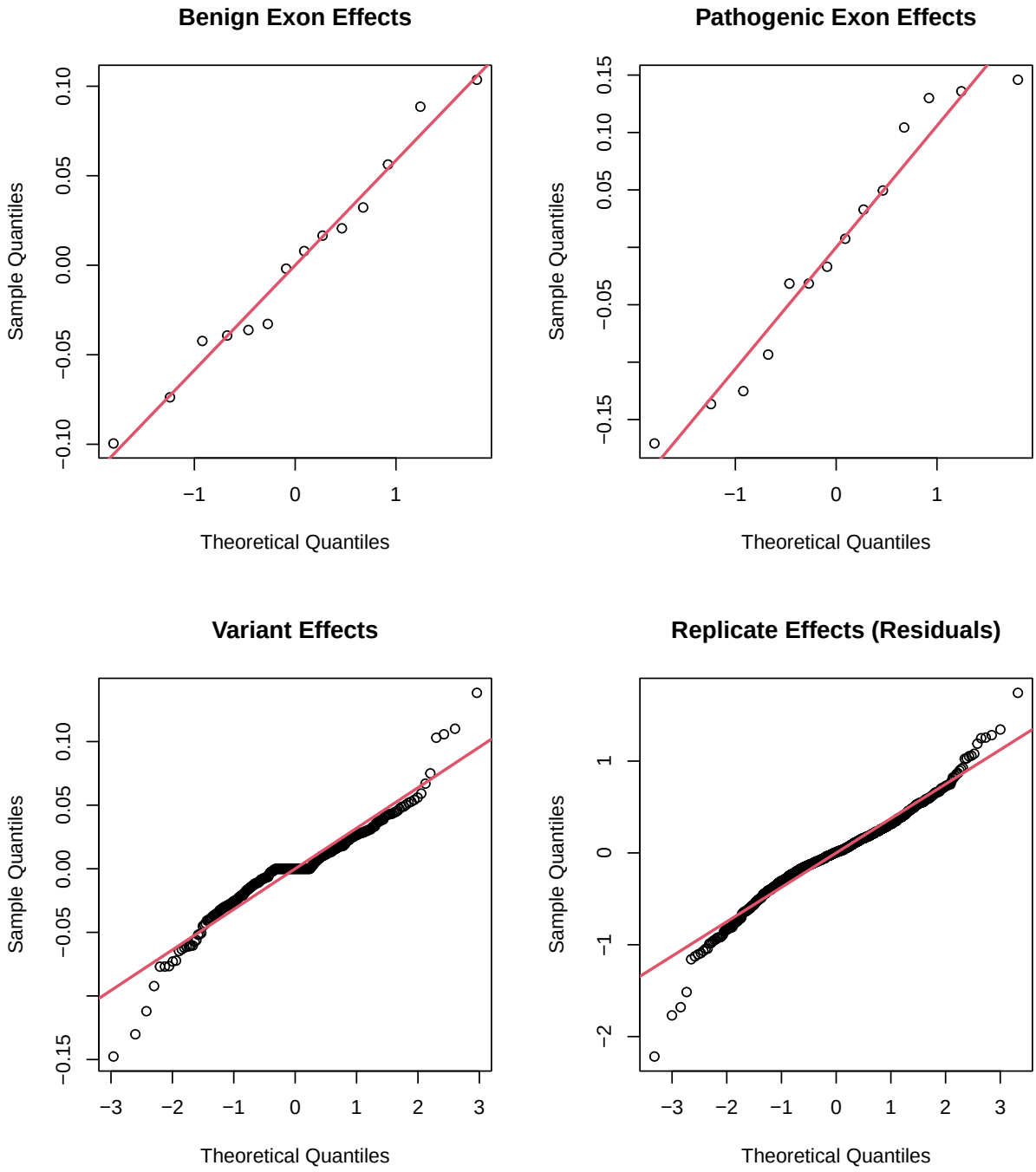
8.2 Mixed Effects Model for the Labeled Variants

Here we fit a linear mixed model using the labeled variants. The model includes a fixed effect for pathogenicity status (`label`) and (nested) random effects for variant and exon and random intercept for label.

```
lme.out2<-lme(lRatio ~ -1 + label, random= ~ -1+label|Exon/variant,data=uvKnown)
summary(lme.out2)

## Linear mixed-effects model fit by REML
##   Data: uvKnown
##        AIC    BIC    logLik
##   1227.234 1272.4 -604.6172
##
## Random effects:
##  Formula: ~-1 + label | Exon
##  Structure: General positive-definite, Log-Cholesky parametrization
##        StdDev     Corr
## labelB 0.07291946 labelB
## labelP 0.14352992 -0.695
##
##  Formula: ~-1 + label | variant %in% Exon
##  Structure: General positive-definite, Log-Cholesky parametrization
##          StdDev     Corr
## labelB   0.09195291 labelB
## labelP   0.25582530 0
## Residual 0.39083496
##
## Fixed effects:  lRatio ~ -1 + label
##             Value  Std.Error  DF    t-value p-value
## labelB  0.1236815 0.02482285 310   4.982566       0
## labelP -0.8447765 0.05710567 310 -14.793215       0
##  Correlation:
##        labelB
## labelP -0.386
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -5.67334457 -0.46789562  0.01385538  0.53373089  4.46055872
##
## Number of Observations: 1119
## Number of Groups:
##              Exon variant %in% Exon
##                14               325

re<-ranef(lme.out2)
names(re)

## [1] "Exon"    "variant"

names(re[[2]])

## [1] "labelB" "labelP"
```

In the following code chunk, we check modeling assumptions.

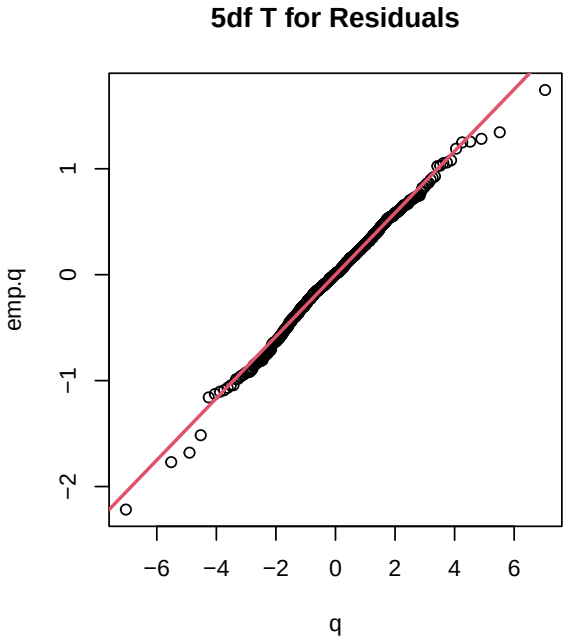
```
par(mfrow=c(2,2))
 qqnorm(re[["Exon"]][,1],main="Benign Exon Effects")
 abline(a=0,b=sd((re[["Exon"]][,1])),lwd=2,col=2)
 qqnorm(re[["Exon"]][,2],main="Pathogenic Exon Effects")
 abline(a=0,b=sd((re[["Exon"]][,2])),lwd=2,col=2)
 qqnorm(re[["variant"]][,1],main="Variant Effects")
 abline(a=0,b=sd((re[["variant"]][,1])),lwd=2,col=2)
 qqnorm(residuals(lme.out2),main="Replicate Effects (Residuals)")
 abline(a=0,b=sd(residuals(lme.out2)),lwd=2,col=2)
```

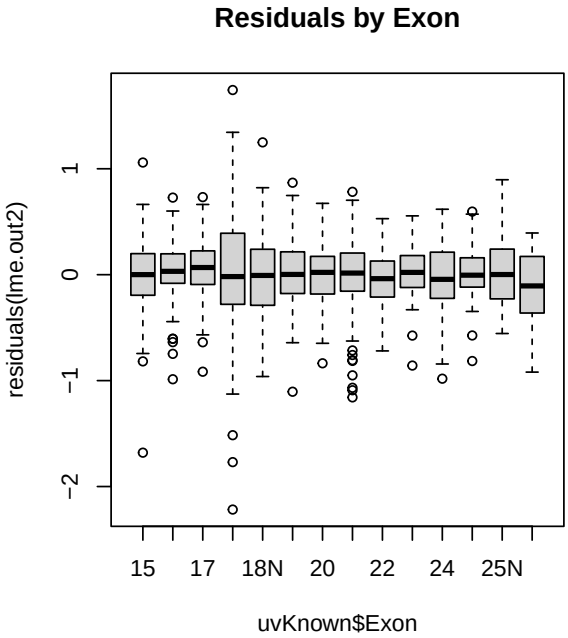


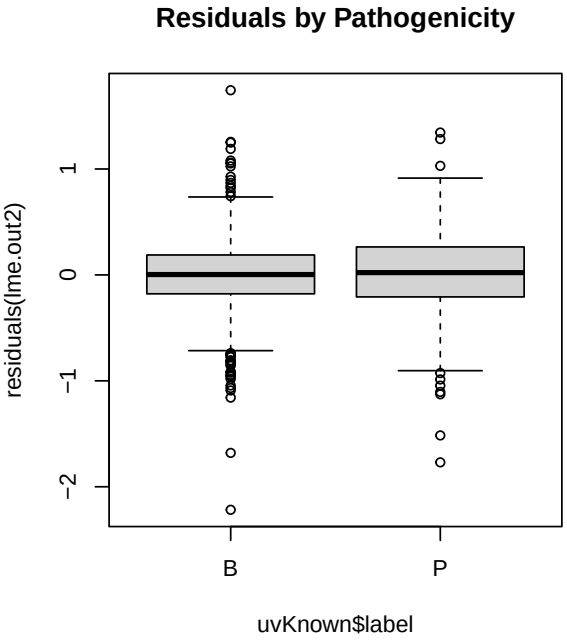
Looks like a t–distribution with approximately five degrees of freedom might be needed for the measurement model:

```
par(mfrow=c(2,2))
q<-qt(p=((1:nrow(uvKnown) - 0.5)/nrow(uvKnown)),df=5)
emp.q<-sort(residuals(lme.out2))
```

```r
plot(q,emp.q,main="5df T for Residuals")
abline(a=0,b=coef(lm(emp.q~-1+q)),lwd=2,col=2)
boxplot(residuals(lme.out2)~uvKnown$Exon,main="Residuals by Exon")
##boxplot(residuals(lme.out2)~uvKnown£p.b,main="Residuals by Pathogenicity")
boxplot(residuals(lme.out2)~uvKnown$label,main="Residuals by Pathogenicity")
##boxplot(residuals(lme.out2)~uvKnown£Exon + uvKnown£p.b,
##        main="Residuals by Exon and Pathogenicity")
```







9 Setup Data Structures for Full VarCall Model

This section creates the data structures needed for the JAGS estimation procedure.

```
##kdel<-as.character(unique(uvMaster£uPOS[uvMaster£P_B=="P"]))
##kneut<-as.character(unique(uvMaster£uPOS[uvMaster£P_B=="B"]))
kdel<-kdel[!is.na(kdel)]
kneut<-kneut[!is.na(kneut)]
length(kdel)
```

```
## [1] 302
```

```
length(kneut)
```

```
## [1] 1285
```

```
known.ids<-c(kdel,kneut)
length(known.ids)
```

```
## [1] 1587
```

```
##table(uvMaster£P_B)
table(uvMaster$label)
```

```
##
##   B   P
## 254  71
```

9.1 Variant Labels

```
########################################
##  Create Structures for JAGS model: ##
########################################
##
## (1) Variant Labels:
##
num.b<-length(unique(uvDB$batch))
uvDB$variant<-factor(uvDB$variant)
##known classifications:
known<-rep(0,length(uvDB$variant))
known[uvDB$variant%in%kdel]<-1     ## known disease associated variant
known[uvDB$variant%in%kneut]<-1    ## known disease neutral variant
uv.ids<-unique(levels(factor(uvDB$variant[known==0])))
length(known.ids)
```

```
## [1] 1587
```

```
length(uv.ids)
```

```
## [1] 1069
```

```
del.known<-rep(NA,length(uvDB$variant))
del.known[uvDB$variant%in%kdel]<-2
del.known[uvDB$variant%in%kneut]<-1
temp<-unique(cbind(as.character(uvDB$variant),as.numeric(uvDB$variant),del.known))
del.known<-temp[,3]
table(del.known)
```

```
## del.known
##   1   2
## 254  71

v.ids<-as.character(temp[,1])
v.codes<-as.numeric(temp[,2])
names(del.known)<-v.ids
names(v.ids)<-v.ids
names(v.codes)<-v.ids
num.v<-length(v.ids)
## Order variant codes 1 to n.var:
temp<-order(v.codes)
v.codes<-v.codes[temp]
v.ids<-v.ids[temp]
del.known<-del.known[temp]
table(v.codes==sort(v.codes))

##
## TRUE
## 1394

table(diff(v.codes))

##
##    1
## 1393

v.codes[1:4]

## 32356417_T_A 32356418_C_T 32356420_T_C 32356422_G_A
##            1            2            3            4
```

10 JAGS Model Estimation and Summary Code

10.1 JAGS Data

The structure `bdat` is a list containing the data needed for fitting the VarCall model using JAGS.

```
num.b<-length(unique(uvDB$batch))
uvDB$variant<-factor(uvDB$variant)
num.v<-length(v.ids)
eta0<-rep(NA,num.v)
alpha.value<-as.numeric(summary(lme.out2)$tTable[1:2,1])
bdat<-list(f.bv=uvDB$lRatio,tdf=t.df,
           batchM=uvDB$batch,
           variantM=as.numeric(uvDB$variant),
           del=as.integer(del.known),
           V=num.v, B=num.b,M=length(uvDB$lRatio),
           eta=eta0,a.pp=beta.a,b.pp=beta.b,
           alpha=alpha.value)
```

10.2 Starting Values

Here we specify a function for generating starting values (`fun.inits()`) and list of model structures (`fun.parameters.bv`) that we would like to 'monitor.' JAGS returns a matrix of samples from the posterior distribution for all variables that are monitored.

```
eta.start<-rnorm(num.v,0,0.25)
beta.start<-rnorm(num.b,0,0.01)
## hist(beta.start)
del.start<-1+(eta.start<(-0.25))
del.start[!is.na(del.known)]<-NA
## Parameters to Monitor
fun.parameters.bv <- c("del" ,
                       "prob", "P",
                       "beta", "mu.beta",
                       "sigma2.beta","gamma","mu.gamma","sigma2.gamma",
                       "eta","sigma2.eta","sigma2.reps")
## Function that sets initial values:
fun.inits <- function(x=bv.leave.off.old) {
    temp<-numeric(length(known.ids))
    names(temp)<-known.ids
    temp[known.ids %in% kdel]<-2
    temp[known.ids %in% kneut]<-1
    return(list(sigma.reps=0.01,
                beta=beta.start,
                sigma.beta=1.45,
                sigma.eta=c(0.50,0.50),
                eta=eta.start,sigma.gamma=0.05,mu.gamma=0.0,
                gamma=exp(rnorm(num.b,mean=0,sd=0.01)),
                mu.beta=0.0,
                del=del.start))
}
```

10.3 Model Specification

The MAVE VarCall model is specified in the external file `maveModel.R`:

```
fun.model.file <- "maveModel.R"
```

10.4 Fit Model

Fit the model using MCMC. **NOTE: this code chunk may take hours to run on a fast numerical server.**

```
jags.out<-jags(data=bdat,
               inits=fun.inits,
               parameters=fun.parameters.bv,
               fun.model.file,
               n.chains=1,
               n.iter=mcmc.pars$iter,
```

```
               n.burnin=mcmc.pars$burn,
               n.thin=mcmc.pars$thin,
               DIC=F,progress.bar="none")

## module glm loaded
## module dic loaded

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5072
##    Unobserved stochastic nodes: 3964
##    Total graph size: 32004
##
## Initializing model
```

10.5 Model Summaries

The matrix of sampled model parameter values is in the structure `jags.out`*BUGSoutput*`sims.matrix`. In the following code chunk, we extract from this matrix blocks of key parameter values. For example, as its name suggests, `eta.samps` contains samples form the posterior distribution (rows) of the variant effect (eta) values for each of the variants (columns).

```
dim(jags.out$BUGSoutput$sims.matrix)

## [1]  500 5680

mpars1<-colnames(jags.out$BUGSoutput$sims.matrix)[substr(colnames(jags.out$BUGSoutput$sims.matrix),1,3)=
mu.samps<-jags.out$BUGSoutput$sims.matrix[,mpars1]
mu.samps<-cbind(rep(bdat$alpha[1],nrow(mu.samps)),
                rep(bdat$alpha[2],nrow(mu.samps)),
                mu.samps)
colnames(mu.samps)[1:2]<-c("alpha[1]","alpha[2]")
rm(mpars1)
var.samps<-jags.out$BUGSoutput$sims.matrix[,(substr(colnames(jags.out$BUGSoutput$sims.matrix),1,5)=="si
beta.samps<-jags.out$BUGSoutput$sims.matrix[,paste("beta[",1:bdat$B,"]",sep="")]
gamma.samps<-jags.out$BUGSoutput$sims.matrix[,paste("gamma[",1:bdat$B,"]",sep="")]
eta.samps<-jags.out$BUGSoutput$sims.matrix[,paste("eta[",1:bdat$V,"]",sep="")]
del.samps<-jags.out$BUGSoutput$sims.matrix[,paste("del[",1:bdat$V,"]",sep="")]
colnames(eta.samps)<-v.ids
colnames(del.samps)<-v.ids
```

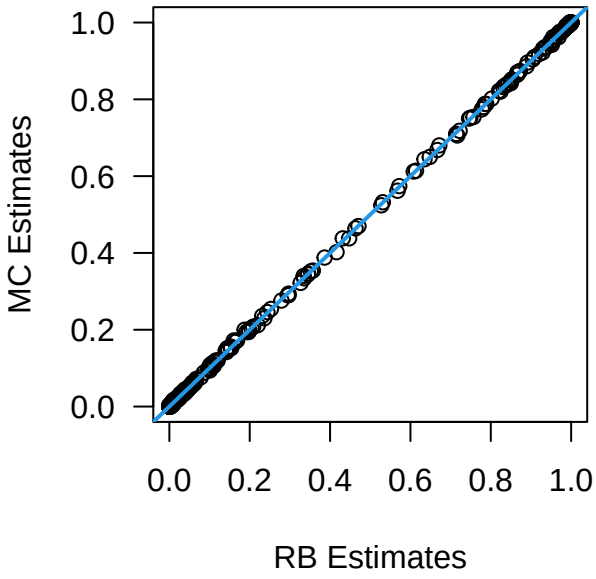
The following code chunk we compute Rao–Blackwellized (RB) estimates of the posterior probabilities that each variant is pathogenic.

```
lpp.threshold<-100
subst<-(!(colnames(del.samps)%in%known.ids))
prdv.samps<-fdv(eta.samps,m.s=mu.samps,
                v.s=var.samps,lpp.cap=lpp.threshold,
                prior.aa=beta.a,prior.bb=beta.b)
names(prdv.samps)
```

```
## [1] "prdv"      "lPostOdds" "lBF"

names(prdv.samps$prdv)<-colnames(eta.samps)
names(prdv.samps$lPostOdds)<-colnames(eta.samps)
names(prdv.samps$lBF)<-colnames(eta.samps)
```

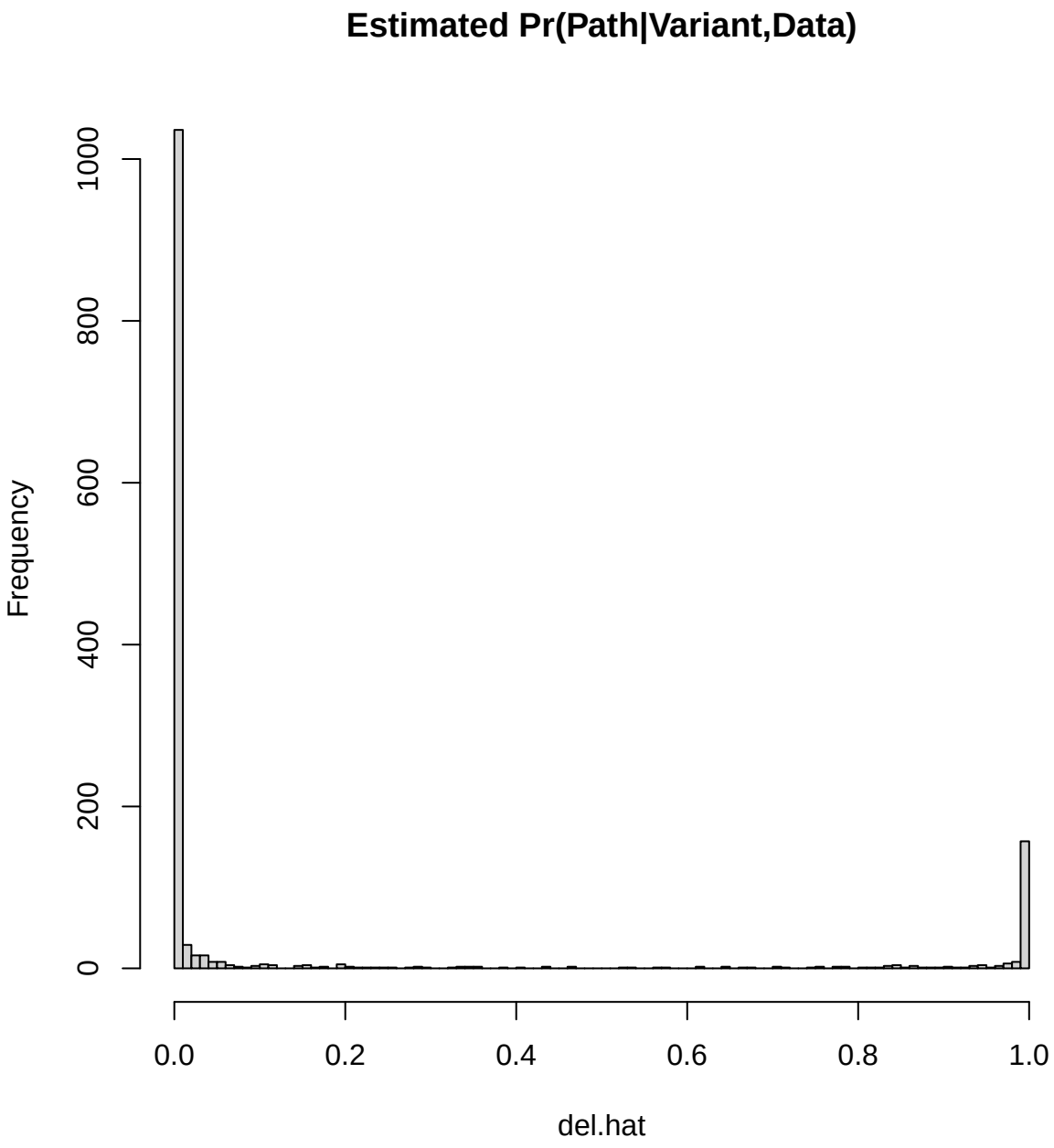
Here we compare the RB estimates to the Monte Carlo averages of the binary variant–specific pathogenicity indicator variables (these two sets of estimates should be very highly correlated):

```
subst<-(!(colnames(del.samps)%in%known.ids))
del.hat<-(apply(del.samps,2,mean) - 1)
cor(prdv.samps$prdv[subst],del.hat[subst])

## [1] 0.9999867

plot(prdv.samps$prdv[subst],del.hat[subst],
     xlab="RB Estimates",ylab="MC Estimates",las=1)
abline(a=0,b=1,col=4,lwd=2)
```

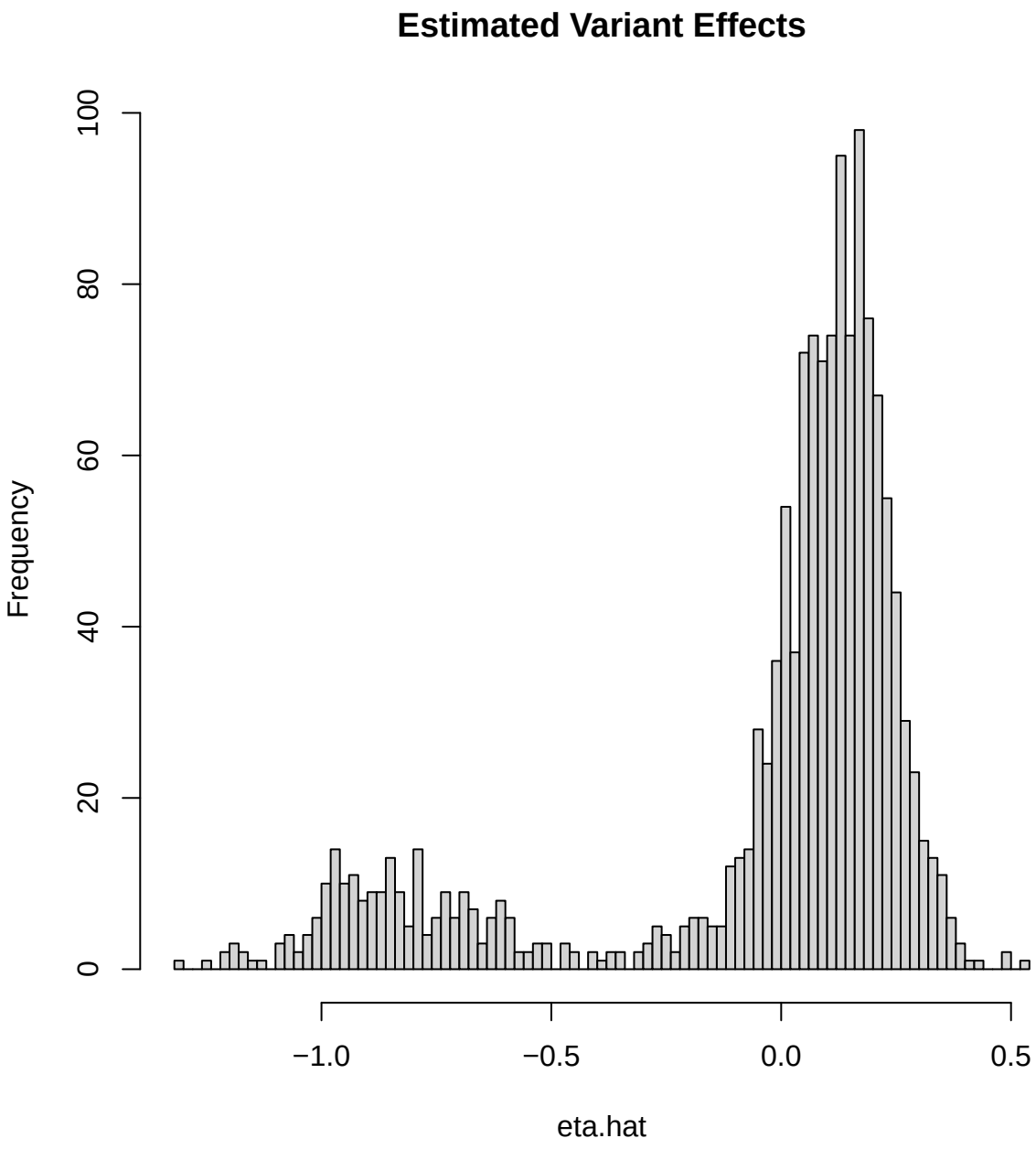


Plot histograms of the variant effects estimates (eta's) and the estimated posterior probabilities of pathogenicity:

```
eta.hat<-(apply(eta.samps,2,mean))
eta.ll<-(apply(eta.samps,2,quantile,probs=0.025))
eta.ul<-(apply(eta.samps,2,quantile,probs=0.975))
hist(del.hat,nclass=100,main="Estimated Pr(Path|Variant,Data)")
```




```
hist(eta.hat,nclass=100,main="Estimated Variant Effects")
```

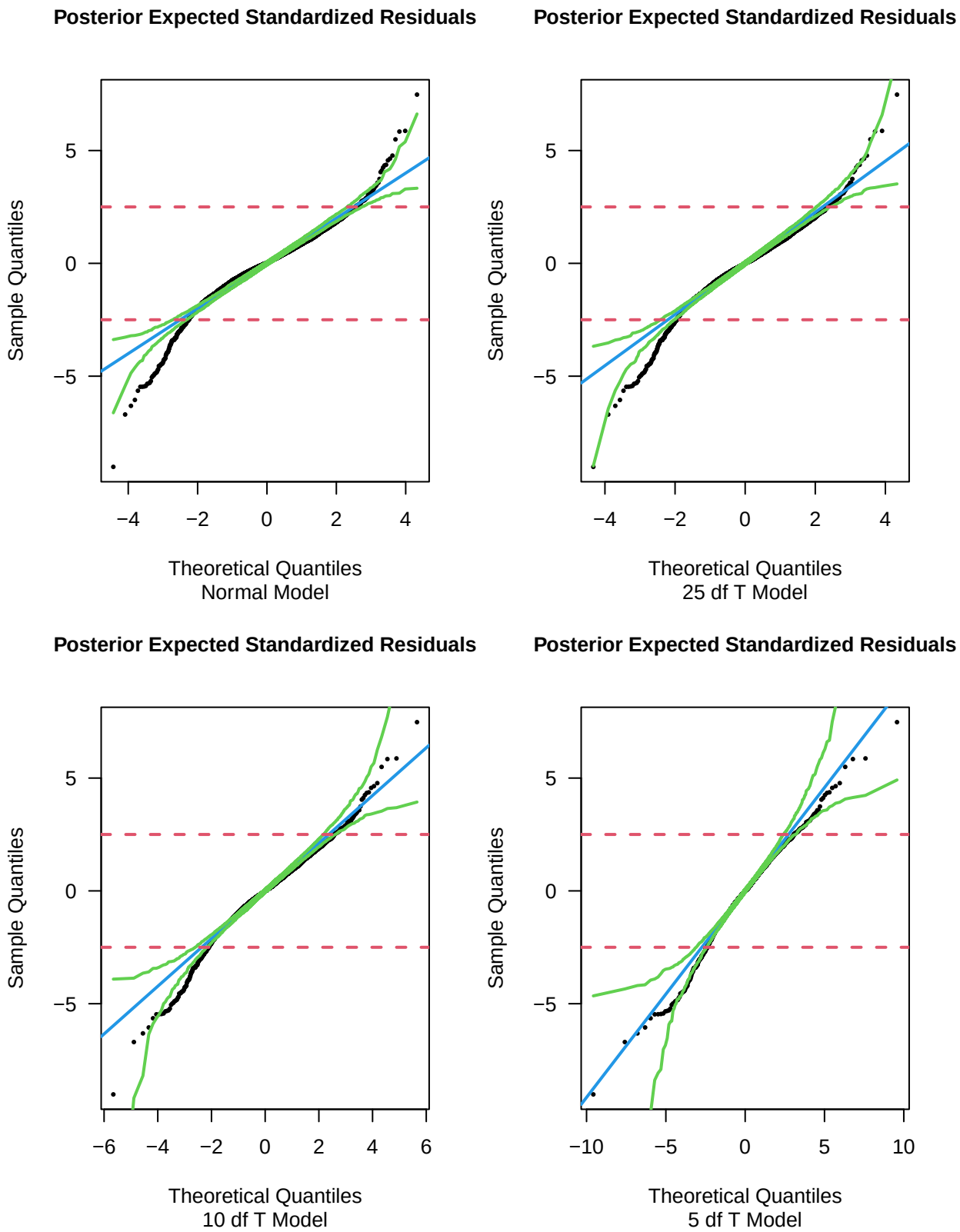


Check the observed distribution of the posterior expected standardized measurement model residuals. The model assumes a five degree of freedom Student's t–distribution. We provide QQ plots here for that specification and three others:

```
mm.batch<-model.matrix(~-1+as.factor(bdat$batchM))
mm.var<-model.matrix(~-1+as.factor(bdat$variantM))
dim(mm.batch)
```

```
## [1] 4747   48
```

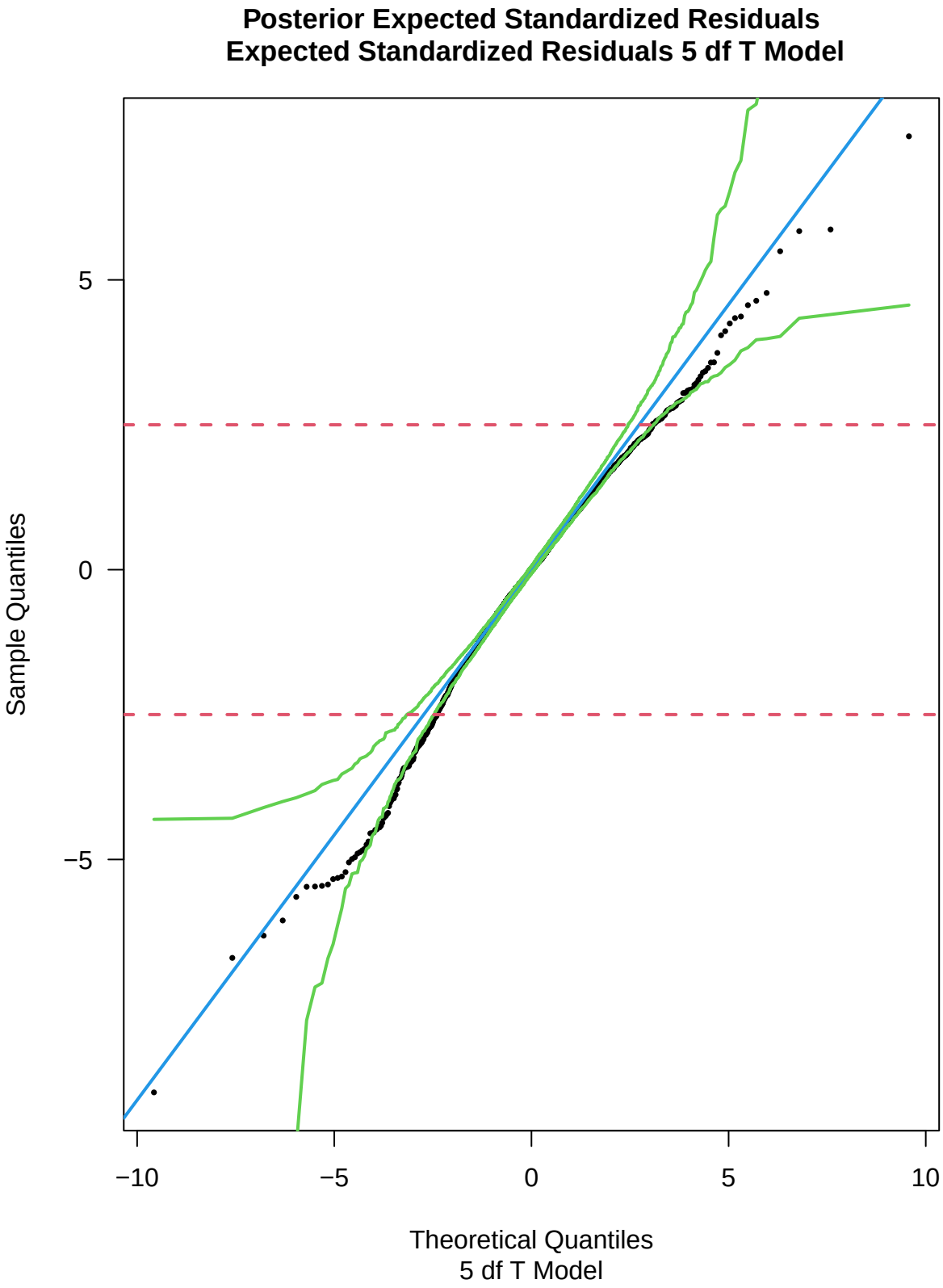
```
dim(mm.var)

## [1] 4747 1394

dim(eta.samps)

## [1]  500 1394

dim(beta.samps)

## [1] 500  48

lin.pred<-((gamma.samps%*%t(mm.batch))*(eta.samps%*%t(mm.var)))+(beta.samps%*%t(mm.batch))
dim(lin.pred)

## [1]  500 4747

lin.pred.v <- sweep(((gamma.samps^2)%*%t(mm.batch)),MARGIN=1,STATS=jags.out$BUGSoutput$sims.matrix[,"si
lin.pred.sd<-sqrt(lin.pred.v)
rm(lin.pred.v,mm.batch,mm.var)
summary(as.numeric(apply(lin.pred.sd,2,mean)))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1516  0.2293  0.2503  0.2756  0.3099  0.5880

expected.resids<-sweep(-lin.pred,MARGIN=2,STATS=bdat$f.bv,FUN="+")
expected.resids<-apply(expected.resids,2,mean)
lin.pred.sd<-apply(lin.pred.sd,2,mean)
expected.stresids<-(expected.resids/lin.pred.sd)
order.er<-order(expected.stresids)
summary(expected.resids)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -3.549060 -0.162740 -0.003393 -0.018538  0.159744  2.102146

lin.pred<-apply(lin.pred,2,mean)
summary(lin.pred)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.13140 -0.07444  0.05660 -0.07821  0.13981  0.78516

par(mfrow=c(2,2))
qqplot2(x=expected.stresids,tdf=101) ##normal qqplot w/ error bars

## NULL

qqplot2(x=expected.stresids,tdf=25) ##t-25 qqplot w/ error bars

## NULL

qqplot2(x=expected.stresids,tdf=10) ##t-10 qqplot w/ error bars

## NULL

qqplot2(x=expected.stresids,tdf=5) ##t-5 qqplot w/ error bars
```


Posterior Expected Standardized Residuals
Sample Quantiles
Theoretical Quantiles
Normal Model
Posterior Expected Standardized Residuals
Sample Quantiles
Theoretical Quantiles
25 df T Model
Posterior Expected Standardized Residuals
Sample Quantiles
Theoretical Quantiles
10 df T Model
Posterior Expected Standardized Residuals
Sample Quantiles
Theoretical Quantiles
5 df T Model

```
## NULL
```

Here is large rendering of the five degree of freedom QQ plot:

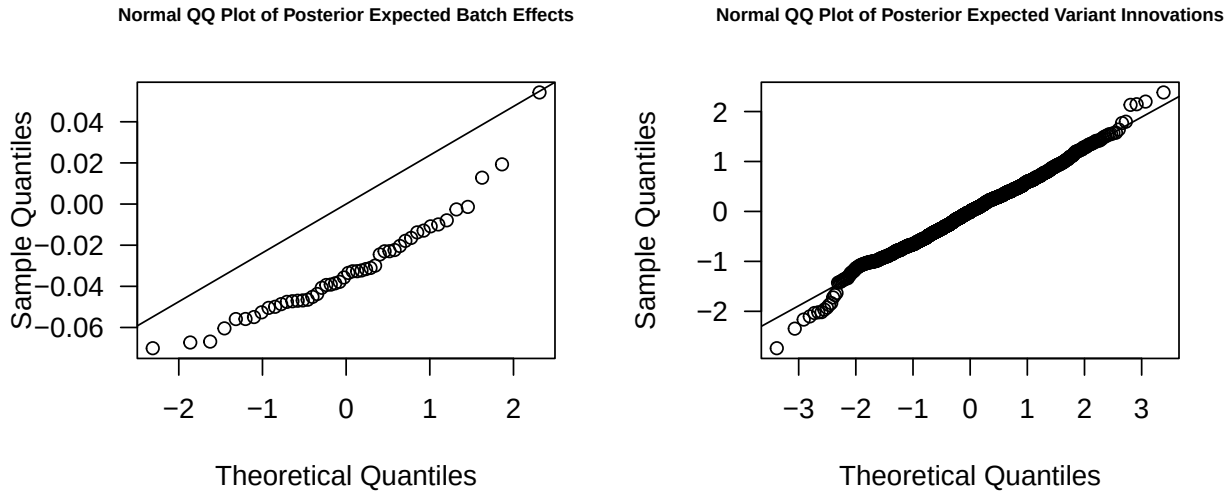
```
qqplot2(x=expected.stresids,tdf=5,
        titl="Expected Standardized Residuals 5 df T Model")
```


Posterior Expected Standardized Residuals
Expected Standardized Residuals 5 df T Model
Sample Quantiles
5
0
−5
−10
−5
0
5
10
Theoretical Quantiles
5 df T Model

```
## NULL
```

Batch mean (beta's) and variant effect (eta's) normal QQ plots (note the beta's are *not* centered at zero):

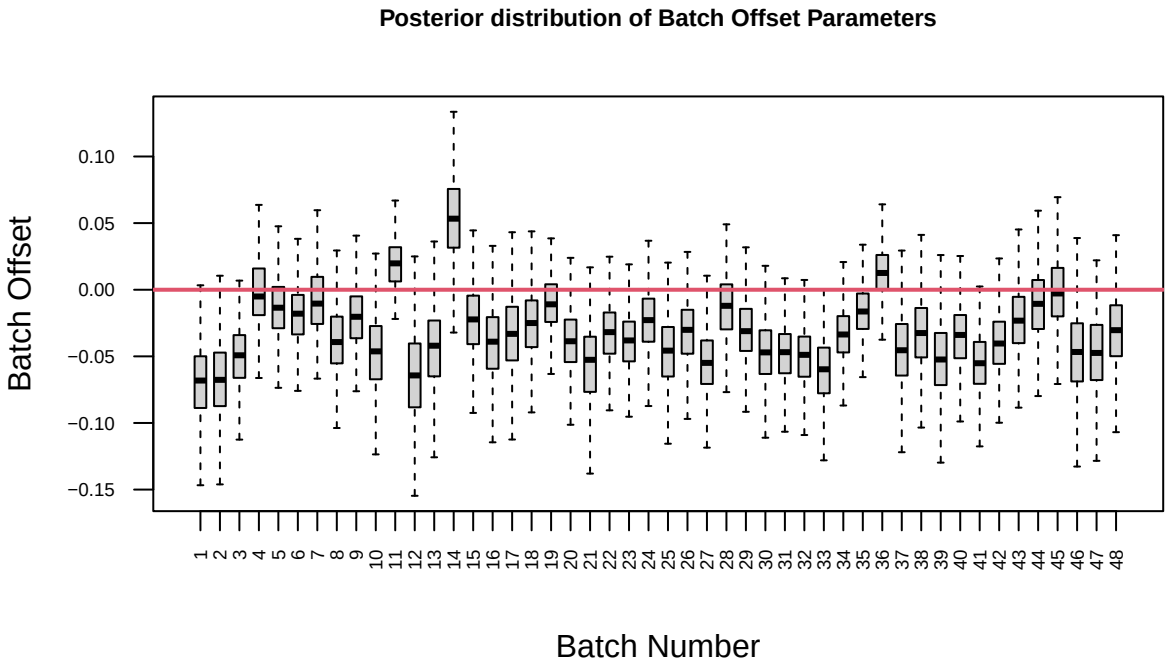
```
## Batch and Variant QQ plots.
par(mfrow=c(1,2))
## (1) batch effects
expected.batch.effects<-apply(beta.samps,2,mean)
sd.batch.effects<-apply(beta.samps,2,sd)
qqnorm(expected.batch.effects,main="Normal QQ Plot of Posterior Expected Batch Effects",las=1,cex.main=0
abline(a=0,b=sd(expected.batch.effects))
## (2) variant effects, need to subtact mixture means, ie calc residuals:
eta.mean<-sweep(1*(del.samps==1),1,STATS=mu.samps[,"alpha[1]"],FUN="*")+sweep(1*(del.samps==2),1,STATS=m
eta.var<-sweep(1*(del.samps==1),1,STATS=var.samps[,"sigma2.eta[1]"],FUN="*")+sweep(1*(del.samps==2),1,ST
eta.resids<-(eta.samps-eta.mean)/sqrt(eta.var)
expected.eta.resids<-apply(eta.resids,2,mean)
qqnorm(expected.eta.resids,main="Normal QQ Plot of Posterior Expected Variant Innovations",las=1,cex.ma
abline(a=0,b=sd(expected.eta.resids))
```

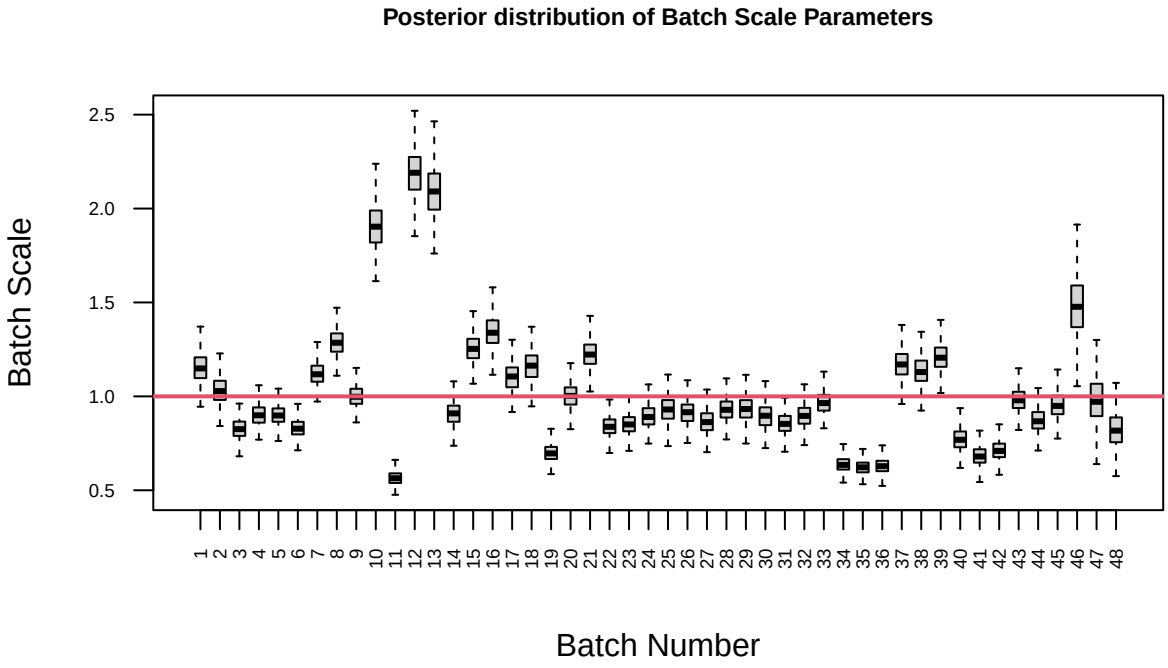


Boxplots of the batch mean (beta's) and scale (gamma's) random effects distributions:

```
batch.dm<-data.frame(batch=rep(1:ncol(beta.samps),each=nrow(beta.samps)),
                     beta=matrix(beta.samps,ncol=1,byrow=F))
par(mfrow=c(2,1))
boxplot(beta~batch,data=batch.dm,las=2,cex.axis=0.60,outline=F,
        pars=list(boxwex=0.6,crt=90),
        main="Posterior distribution of Batch Offset Parameters",
        ylab="Batch Offset",xlab="Batch Number",cex.main=0.75)
abline(h=0,lwd=2,col=2)
gamma.dm<-data.frame(batch=rep(1:ncol(gamma.samps),each=nrow(gamma.samps)),
                     gamma=matrix(gamma.samps,ncol=1,byrow=F))
boxplot(gamma~batch,data=gamma.dm,las=2,cex.axis=0.60,outline=F,
```

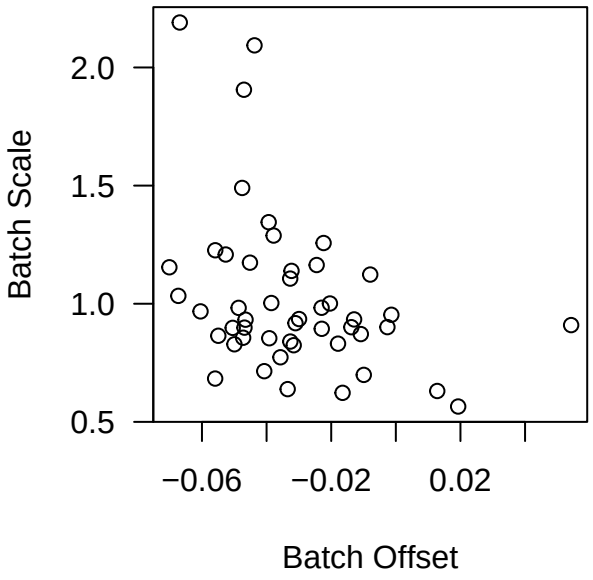
```
        pars=list(boxwex=0.6,crt=90),
        main="Posterior distribution of Batch Scale Parameters",
        ylab="Batch Scale",xlab="Batch Number",cex.main=0.75)
abline(h=1,lwd=2,col=2)
```





A scatter plot of posterior mean batch means vers batch scalings:

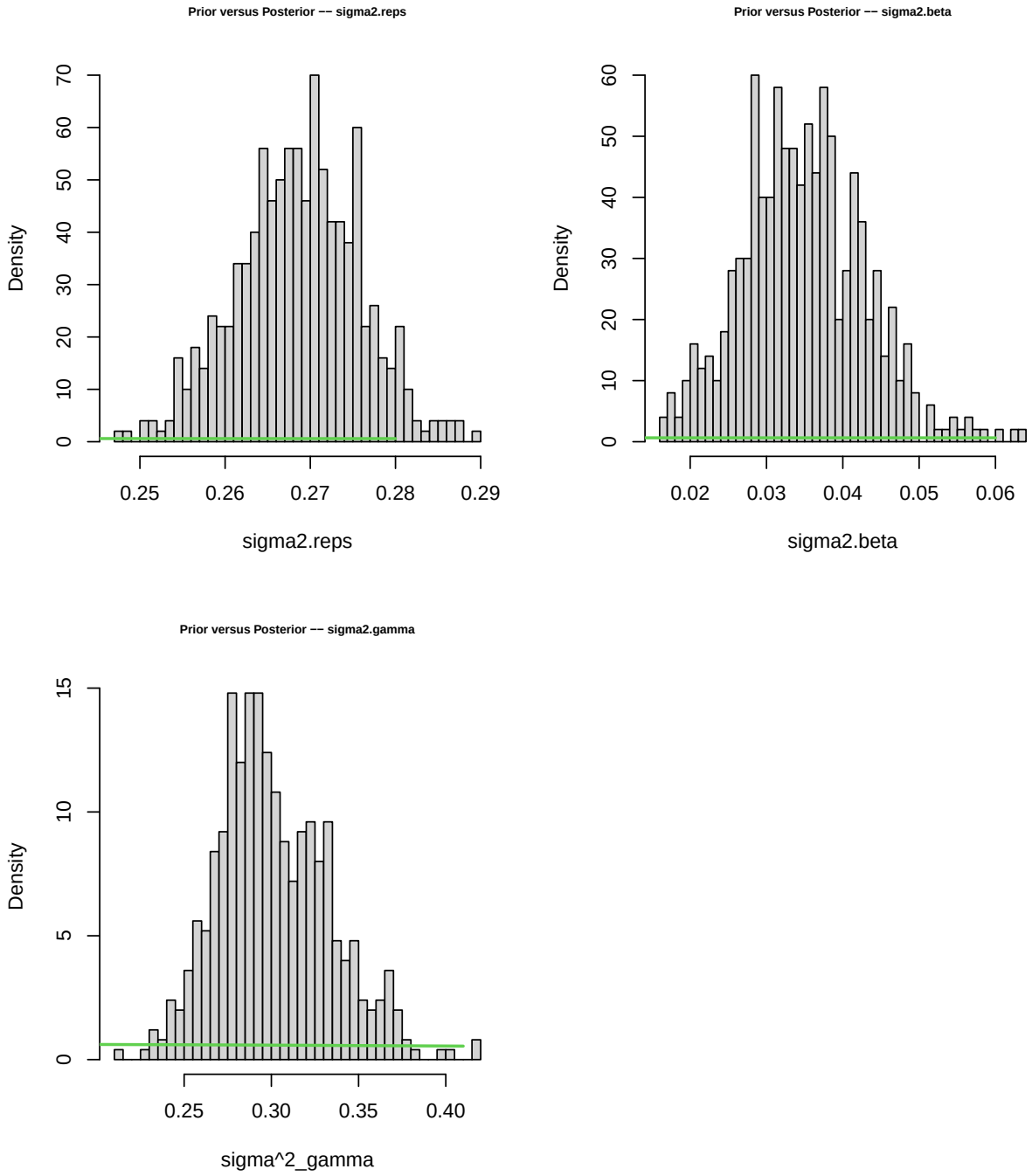
```
plot(apply(beta.samps,2,mean),apply(gamma.samps,2,mean),las=1,
     xlab="Batch Offset",ylab="Batch Scale")
```



10.6 Graphical Summaries of Parameter Estimates

Histograms of the marginal posterior distributions of the variance components: the measurment error squared scale parameter `sigma2.reps`, the variance of the batch mean random effects `sigma2.beta` and the variance of the logged batch scale random effects `sigma2.gamma`.

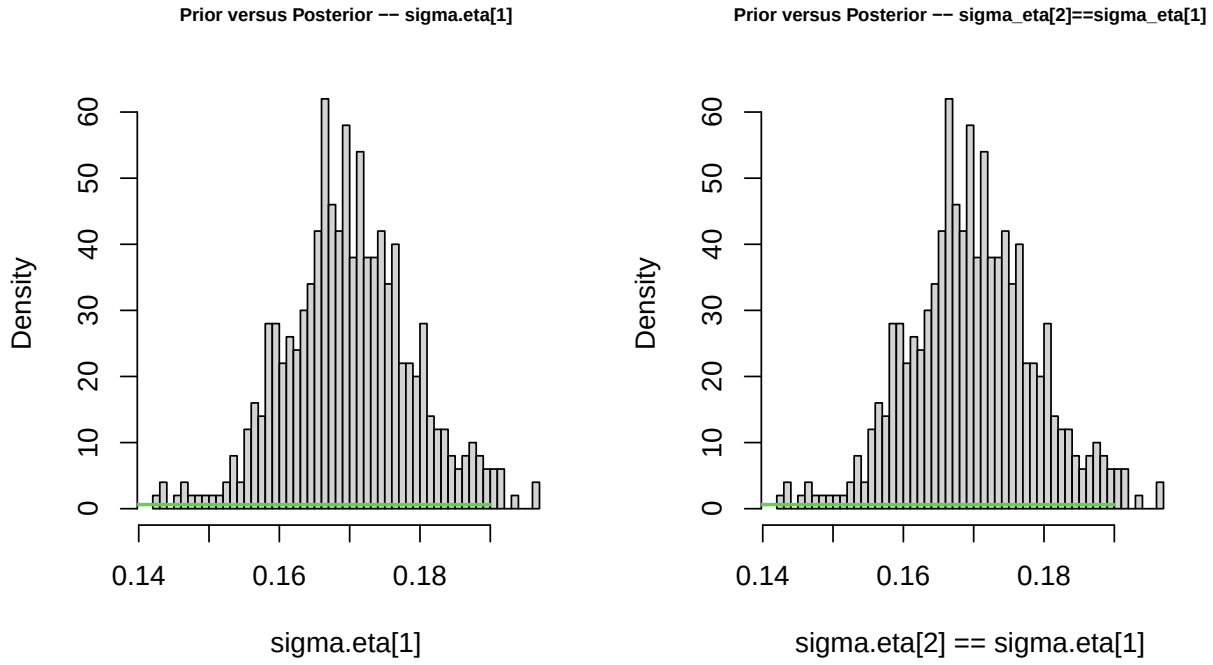
```
par(mfrow=c(2,2))
## sigma2.reps
hist(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.reps"]),cex.main=0.65,
     prob=T,nclass=50,xlab="sigma2.reps",main="Prior versus Posterior -- sigma2.reps")
lines(grid<-seq(0,max(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.reps"])),by=0.01),
      2*dt(grid,df=1),col=3,lwd=2)
## sigma2.beta
hist(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.beta"]),cex.main=0.65,
     prob=T,nclass=50,xlab="sigma2.beta",main="Prior versus Posterior -- sigma2.beta")
lines(grid<-seq(0,max(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.beta"])),by=0.01),
      2*dt(grid,df=1),col=3,lwd=2)
## sigma2.gamma
hist(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.gamma"]),cex.main=0.65,
     prob=T,nclass=50,xlab="sigma^2_gamma",main="Prior versus Posterior -- sigma2.gamma")
lines(grid<-seq(0,max(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.gamma"])),by=0.01),
      2*dt(grid,df=1),col=3,lwd=2)
```



Histograms of the marginal posterior distributions of the component–wise variances (they are constrained to be equal) for the two–component normal mixture model for the variant effects (eta's).

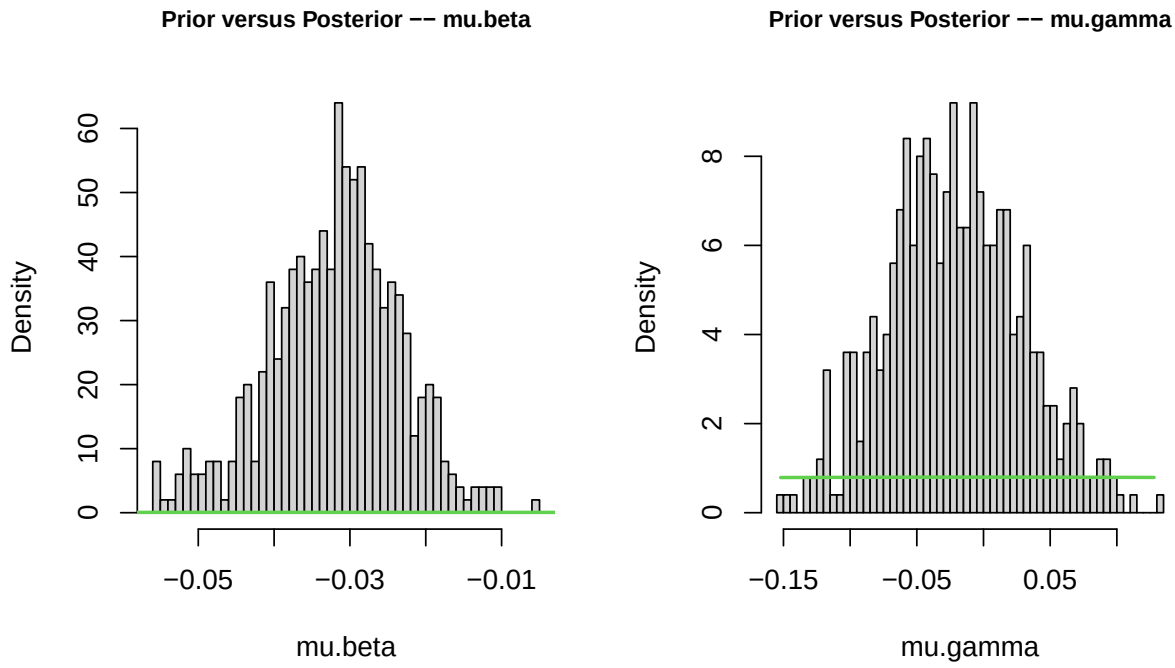
```
par(mfrow=c(1,2))
## sigma2.eta[1]
hist(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.eta[1]"]),cex.main=0.65,
```

```
     prob=T,nclass=50,xlab="sigma.eta[1]",main="Prior versus Posterior -- sigma.eta[1]")
lines(grid<-seq(0,max(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.eta[1]"])),by=0.01),
      2*dt(grid,df=1),col=3,lwd=2)
## sigma2.eta[2]
hist(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.eta[2]"]),cex.main=0.65,
     prob=T,nclass=50,xlab="sigma.eta[2] == sigma.eta[1]",
     main="Prior versus Posterior -- sigma_eta[2]==sigma_eta[1]")
lines(grid<-seq(0,max(sqrt(jags.out$BUGSoutput$sims.matrix[,"sigma2.eta[2]"])),by=0.01),
      2*dt(grid,df=1),col=3,lwd=2)
```



Histograms of the marginal posterior distributions of the mean of the batch mean parameters (`mu.beta`) and the mean of the logged batch scale parameters (`mu.gamma`).

```
##Mean parameters:
par(mfrow=c(1,2))
## mu.beta
hist(jags.out$BUGSoutput$sims.matrix[,"mu.beta"],cex.main=0.85,
     prob=T,nclass=50,xlab="mu.beta",
     main="Prior versus Posterior -- mu.beta")
abline(h=(1/20),col=3,lwd=2)
## mu.gamma
hist(jags.out$BUGSoutput$sims.matrix[,"mu.gamma"],cex.main=0.85,
     prob=T,nclass=50,xlab="mu.gamma",
     main="Prior versus Posterior -- mu.gamma")
lines(grid<-seq(min(jags.out$BUGSoutput$sims.matrix[,"mu.gamma"]),
                max(jags.out$BUGSoutput$sims.matrix[,"mu.gamma"]),by=0.01),
     2*dnorm(grid,mean=0,sd=1),col=3,lwd=2)
```



10.7 Table of Estimated Variant Effects

Assemble a table of variant–level results for export:

```
pr.del<-prdv.samps$prdv
length(pr.del)

## [1] 1394

pr.del[1:3]

## 32356417_T_A 32356418_C_T 32356420_T_C
## 4.273412e-03 1.238982e-02 1.461764e-05

tabl<-annot
colnames(tabl)[2]<-"variant"
dim(tabl)

## [1] 1394   25

length(unique(tabl$variant))

## [1] 1394

rownames(tabl)<-tabl$variant
tabl<-data.frame(tabl)
tabl$PrDel<-rep(NA,nrow(tabl))
table(names(pr.del) %in% rownames(tabl))
```

```
##
## TRUE
## 1394
```

```
tabl[names(pr.del),"PrDel"]<-pr.del
table(tabl$P_B[is.na(tabl$PrDel)])
```

```
## < table of extent 0 >
```

```
table(tabl$P_B[!is.na(tabl$PrDel)])
```

```
##
##  B  P
## 59 56
```

```
lp.odds<-prdv.samps$lPostOdds
tabl$lPostOdds<-rep(NA,nrow(tabl))
table(names(lp.odds) %in% rownames(tabl))
```

```
##
## TRUE
## 1394
```

```
tabl[names(lp.odds),"lPostOdds"]<-lp.odds
lbf<-prdv.samps$lBF
tabl$logBF<-rep(NA,nrow(tabl))
table(names(lbf) %in% rownames(tabl))
```

```
##
## TRUE
## 1394
```

```
tabl[names(lbf),"logBF"]<-lbf
tabl$Status<-rep("VUS",nrow(tabl))
tabl$Status[rownames(tabl)%in%kdel]<-"Path"
tabl$Status[rownames(tabl)%in%kneut]<-"Neut"
table(tabl$Status)
```

```
##
## Neut Path  VUS
##  254   71 1069
```

```
temp<-apply(eta.samps,2,mean,na.rm=TRUE)
temp<-temp[rownames(tabl)]
table(names(temp) == rownames(tabl))
```

```
##
## TRUE
## 1394
```

```
tabl$eta <- temp; rm(temp)
eta.ll<-(apply(eta.samps,2,quantile,probs=0.025))
eta.ul<-(apply(eta.samps,2,quantile,probs=0.975))
tabl$eta.ll<-rep(NA,nrow(tabl))
tabl[names(eta.ll),"eta.ll"]<-eta.ll
tabl$eta.ul<-rep(NA,nrow(tabl))
tabl[names(eta.ul),"eta.ul"]<-eta.ul
```

```
## Output Table of Estimates:
write.table(tabl,file="MAVEpostProbs.csv",quote=FALSE,sep=",",row.names=FALSE,col.names=TRUE)
```

11 Plots of Supplemental Materials

These are individual plots for use in the supplement:

```
pdf("ResidualQQ.pdf",width=8,height=8)
qqplot2(x=expected.stresids,tdf=5,
        titl="Expected Standardized Residuals 5 df T Model")

## NULL

dev.off()

## pdf
##   2

pdf("PrDelHistogram.pdf",width=8,height=8)
hist(del.hat,nclass=100,
     main="Estimated Pr(Path | Variant, Data)\n All Assayed Variants",
     las=1)
dev.off()

## pdf
##   2

pdf("EtaHistogram.pdf",width=8,height=6)
hist(eta.hat,nclass=100,cex.main=1.4,
     main="Estimated Variant Effects (Eta's)\n All Assayed Variants",
     las=1)
dev.off()

## pdf
##   2

pdf("BatchMeanQQ.pdf",width=8,height=8)
qqplot2(expected.batch.effects,tdf=1000,
       titl="Normal QQ Plot of Posterior Expected Batch Mean Effects")

## NULL

dev.off()

## pdf
##   2

batch.scale.effects<-log(apply(gamma.samps,2,mean))
pdf("BatchScaleQQ.pdf",width=8,height=8)
qqplot2(batch.scale.effects,tdf=1000,
       titl="Normal QQ Plot of Posterior Logged Expected Scale Effects")
```

```
## NULL

dev.off()

## pdf
##   2

pdf("VariantQQ.pdf",width=8,height=8)
qqplot2(expected.eta.resids,tdf=1000,
        titl="Normal QQ Plot of Posterior Expected Mean--Adjusted Variant Innovations")

## NULL

dev.off()

## pdf
##   2

pdf("BatchMeanBPlot.pdf",width=10,height=6)
boxplot(beta~batch,data=batch.dm,las=2,outline=F,
         pars=list(boxwex=0.6,crt=90),
        main="Posterior distribution of Batch Mean Parameters `Beta'",
        ylab="Batch Mean",xlab="Batch Number",cex.main=1.5)
abline(h=0,lwd=2,col=2)
dev.off()

## pdf
##   2

pdf("BatchScaleBPlot.pdf",width=10,height=6)
boxplot(gamma~batch,data=gamma.dm,las=2,cex.axis=0.60,outline=F,
        pars=list(boxwex=0.6,crt=90),
        main="Posterior distribution of Batch Scale Parameters `Gamma'",
        ylab="Batch Scale",xlab="Batch Number",cex.main=1.5)
abline(h=1,lwd=2,col=2)
dev.off()

## pdf
##   2

pdf("BatchLocationVsScale.pdf",width=8,height=8)
plot(apply(beta.samps,2,mean),apply(gamma.samps,2,mean),las=1,
     main="Scatter Plot of Batch Mean Versus Scale Adjustments",
     xlab="Batch Mean",ylab="Batch Scale",pch=16)
dev.off()

## pdf
##   2
```

12 Wrap–Up

```r
gc(); save.image()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2390219 127.7    4291001  229.2   4291001  229.2
## Vcells 18365151 140.2  162272240 1238.1 264259029 2016.2
```