AnaCoDa_Fall2024_Garcia

Sarah Garcia

2024-09-17

Objective:

- Determine if the mutation rate is different between the forward and reverse strands of bacteria.
- Using E.coli K-12 genome for testing (obtained via Ensembl Bacteria)

Using AnaCoDa

Load needed libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(AnaCoDa)
```

```
## Loading required package: Rcpp
## Loading required package: VGAM
## Loading required package: stats4
## Loading required package: splines
## Loading required package: mvtnorm
```

```
library(seqinr)
```

```
##
## Attaching package: 'seqinr'
##
## The following object is masked from 'package:dplyr':
##
##     count
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.11.1
## - Online documentation and vignettes at mc-stan.org/bayesplot
## - bayesplot theme set to bayesplot::theme_default()
##    * Does _not_ affect other ggplot2 plots
##    * See ?bayesplot_theme_set for details on theme setting
```

Load needed functions - long code chunk, minimize for ease of viewing after loading in functions

```
#Define function
chomp<-function(dir){

  genome <- read_delim(dir, delim = "\n", col_names = FALSE, col_types = "c")
  genome.list <- genome %>% na.omit() %>% data.frame()

  i=0                                              #Initialize empty objects for faster processing time
  flag=0
  y<-data.frame()
  gene.length<-list()
  sequence.all<-list()
  alist<-list()
  #species.name<-species.list                       #this will be user supplied, must match how it looks
  b<-''
  c<-''

  cat(paste("Now importing:",dir, "\n"))          #basic progress checker
  for(i in genome.list[ ,1]){                    #for each line of inputted genomeFASTA
    if(str_detect(i,"^>")){                       #if row starts with >
      if(flag==1){                                #flag check to concatenate sequence lines
        sequence<-str_c(alist,collapse = "")      #collapse
        aasize<-nchar(sequence)                   #get amino acid size
        gene.length<-rbind.data.frame(gene.length,aasize)  #store size
        sequence.all<-rbind.data.frame(sequence.all,sequence)  #store sequence
        alist<-list()                             #empty temp variable
        flag=0                                    #reset flag
      }
      flag=0                                      #reset
      pattern.chr<-paste0("Chromosome:\\S+")#make species name pattern
      pattern.gn<-paste0("description:.*")             #make gene name pattern
      pattern.type<-paste0("gene_biotype:\\S+")            #make >tr or >sp pattern
      a<-str_extract(i,pattern = pattern.chr) #find and store species name (user supplies species name)
      b<-str_extract(i,pattern = pattern.gn)    #find and store gene name
      c<-str_extract(i,pattern = pattern.type)  #find and store pattern type (>tr or >sp, should be onl
      abci<-c(a,b,c,i)                             #make above identifiers into easy to bind object
      y<-rbind(y,abci)                            #rbind to fill rows of new df
    }
    else{
      flag=1                                      #Raise the flag!
      b<-str_trim(i,side = c("right"))            #Cut off new line character
      alist[[i]]<-b                               #Storing all sequence lines of current protein
    }
```

```
  }
  if(flag==1){                                    #this catches the last protein's sequence VVV
    sequence<-str_c(alist,collapse = "")          #
    aasize<-nchar(sequence)                       #
    gene.length<-rbind.data.frame(gene.length,aasize) #
    sequence.all<-rbind.data.frame(sequence.all,sequence) #
    alist<-list()                                 #
    flag=0                                        #^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  }
  y[,5]<-gene.length                               #add sizes to df
  y[,6]<-sequence.all                              #add sequences to df
  colnames(y)<-c("Location","Description","Type","Header","Length","Sequence") #add colnames to object
  y$Description <- str_replace_all(y$Description, "description:", "")  #remove OS= from species column
  y$Location <- str_replace_all(y$Location, "Chromosome:", "")  #remove GN= from gene name column
  y$Type <- str_replace_all(y$Type, "gene_biotype:", "")
  y$Header <- str_replace_all(y$Header, ">", "")

  #Count the number of 'genes' that don't have descriptions
  perc <- (sum(is.na(y$Description)))/nrow(y)*100
  cat("Percentage of individual sequences w/o description:", perc)

  genome <<- y
}   #Load 'chomp' function to import .fasta files

#load("parameter_out_Ecoli_split_09112024.Rda")
loadROCParameterObject <- function(parameter, files){

    setBaseInfo <- function(parameter, files){
      for (i in 1:length(files)) {
        tempEnv <- new.env();
        load(file = files[i], envir = tempEnv)
        if (i == 1) {
          categories <- tempEnv$paramBase$categories
          categories.matrix <- do.call("rbind", tempEnv$paramBase$categories)
          numMixtures <- tempEnv$paramBase$numMix
          numMutationCategories <- tempEnv$paramBase$numMut
          numSelectionCategories <- tempEnv$paramBase$numSel
          mixtureAssignment <- tempEnv$paramBase$curMixAssignment
          lastIteration <- tempEnv$paramBase$lastIteration
          max <- tempEnv$paramBase$lastIteration + 1
          grouplist <- tempEnv$paramBase$grouplist

          stdDevSynthesisRateTraces <- vector("list", length = numSelectionCategories)
          for (j in 1:numSelectionCategories) {
            stdDevSynthesisRateTraces[[j]] <- tempEnv$paramBase$stdDevSynthesisRateTraces[[j]][1:max]
          }
          stdDevSynthesisRateAcceptanceRateTrace <- tempEnv$paramBase$stdDevSynthesisRateAcceptRatTrace
          synthesisRateTrace <- vector("list", length = numSelectionCategories)
          for (j in 1:numSelectionCategories) {
            for (k in 1:length(tempEnv$paramBase$synthRateTrace[[j]])){
              synthesisRateTrace[[j]][[k]] <- tempEnv$paramBase$synthRateTrace[[j]][[k]][1:max]
            }
          }
```

```
    synthesisRateAcceptanceRateTrace <- tempEnv$paramBase$synthAcceptRatTrace
    mixtureAssignmentTrace <- vector("list", length = length(tempEnv$paramBase$mixAssignTrace))
    for (j in 1:length(tempEnv$paramBase$mixAssignTrace)){
      mixtureAssignmentTrace[[j]] <- tempEnv$paramBase$mixAssignTrace[[j]][1:max]
    }
    mixtureProbabilitiesTrace <- c()
    for (j in 1:numMixtures) {
      mixtureProbabilitiesTrace[[j]] <- tempEnv$paramBase$mixProbTrace[[j]][1:max]
    }
    codonSpecificAcceptanceRateTrace <- tempEnv$paramBase$codonSpecificAcceptRatTrace

    ### ERROR HERE ###
    withPhi <- tempEnv$paramBase$withPhi
    if (withPhi){
      phiGroups <- length(tempEnv$paramBase$synthesisOffsetTrace) #add $paramBase
      synthesisOffsetTrace <- c()
      for (j in 1:phiGroups) {
        synthesisOffsetTrace[[j]] <- tempEnv$paramBase$synthesisOffsetTrace[[j]][1:max]
      }


      synthesisOffsetAcceptanceRateTrace <- tempEnv$paramBase$synthesisOffsetAcceptRatTrace


      observedSynthesisNoiseTrace <- c()
      for (j in 1:phiGroups) {
        observedSynthesisNoiseTrace[[j]] <- tempEnv$paramBase$observedSynthesisNoiseTrace[[j]][1:r
      }
      #need number of phi groups, not the number of mixtures apparently.
    }else {
      synthesisOffsetTrace <- c()
      synthesisOffsetAcceptanceRateTrace <- c()
      observedSynthesisNoiseTrace <- c()
    }
  } else {
    if (sum(categories.matrix != do.call("rbind", tempEnv$paramBase$categories)) != 0){
      stop("categories is not the same between all files")
    }#end of error check

    if (numMixtures != tempEnv$paramBase$numMix){
      stop("The number of mixtures is not the same between files")
    }

    if (numMutationCategories != tempEnv$paramBase$numMut){
      stop("The number of mutation categories is not the same between files")
    }

    if (numSelectionCategories != tempEnv$paramBase$numSel){
      stop("The number of selection categories is not the same between files")
    }

    if (length(mixtureAssignment) != length(tempEnv$paramBase$curMixAssignment)){
      stop("The length of the mixture assignment is not the same between files.
```

```
            Make sure the same genome is used on each run.")
      }

      if(length(grouplist) != length(tempEnv$paramBase$grouplist)){
        stop("Number of Amino Acids/Codons is not the same between files.")
      }
      if (withPhi != tempEnv$paramBase$withPhi){
        stop("Runs do not match in concern in with.phi")
      }

      curSynthesisOffsetTrace <- tempEnv$paramBase$synthesisOffsetTrace
      curSynthesisOffsetAcceptanceRateTrace <- tempEnv$paramBase$synthesisOffsetAcceptRatTrace
      curObservedSynthesisNoiseTrace <- tempEnv$paramBase$observedSynthesisNoiseTrace

      if (withPhi){
        combineTwoDimensionalTrace(synthesisOffsetTrace, curSynthesisOffsetTrace, max)
        size <- length(curSynthesisOffsetAcceptanceRateTrace)
        combineTwoDimensionalTrace(synthesisOffsetAcceptanceRateTrace, curSynthesisOffsetAcceptanceR
        combineTwoDimensionalTrace(observedSynthesisNoiseTrace, curObservedSynthesisNoiseTrace, max)
      }


      curStdDevSynthesisRateTraces <- tempEnv$paramBase$stdDevSynthesisRateTraces
      curStdDevSynthesisRateAcceptanceRateTrace <- tempEnv$paramBase$stdDevSynthesisRateAcceptRatTra
      curSynthesisRateTrace <- tempEnv$paramBase$synthRateTrace
      curSynthesisRateAcceptanceRateTrace <- tempEnv$paramBase$synthAcceptRatTrace
      curMixtureAssignmentTrace <- tempEnv$paramBase$mixAssignTrace
      curMixtureProbabilitiesTrace <- tempEnv$paramBase$mixProbTrace
      curCodonSpecificAcceptanceRateTrace <- tempEnv$paramBase$codonSpecificAcceptRatTrace

      lastIteration <- lastIteration + tempEnv$paramBase$lastIteration


      #assuming all checks have passed, time to concatenate traces
      max <- tempEnv$paramBase$lastIteration + 1
      combineTwoDimensionalTrace(stdDevSynthesisRateTraces, curStdDevSynthesisRateTraces, max)

      size <- length(curStdDevSynthesisRateAcceptanceRateTrace)
      stdDevSynthesisRateAcceptanceRateTrace <- c(stdDevSynthesisRateAcceptanceRateTrace,
                                                  curStdDevSynthesisRateAcceptanceRateTrace[2:size])


      combineThreeDimensionalTrace(synthesisRateTrace, curSynthesisRateTrace, max)
      size <- length(curSynthesisRateAcceptanceRateTrace)
      combineThreeDimensionalTrace(synthesisRateAcceptanceRateTrace, curSynthesisRateAcceptanceRate

      combineTwoDimensionalTrace(mixtureAssignmentTrace, curMixtureAssignmentTrace, max)
      combineTwoDimensionalTrace(mixtureProbabilitiesTrace, curMixtureProbabilitiesTrace, max)
      size <- length(curCodonSpecificAcceptanceRateTrace)
      combineTwoDimensionalTrace(codonSpecificAcceptanceRateTrace, curCodonSpecificAcceptanceRateTra
    }
  }
```

```
    parameter$setCategories(categories)
    parameter$setCategoriesForTrace()
    parameter$numMixtures <- numMixtures
    parameter$numMutationCategories <- numMutationCategories
    parameter$numSelectionCategories <- numSelectionCategories
    parameter$setMixtureAssignment(tempEnv$paramBase$curMixAssignment) #want the last in the file seq
    parameter$setLastIteration(lastIteration)
    parameter$setGroupList(grouplist)

    trace <- parameter$getTraceObject()
    trace$setStdDevSynthesisRateTraces(stdDevSynthesisRateTraces)
    trace$setStdDevSynthesisRateAcceptanceRateTrace(stdDevSynthesisRateAcceptanceRateTrace)
    trace$setSynthesisRateTrace(synthesisRateTrace)
    trace$setSynthesisRateAcceptanceRateTrace(synthesisRateAcceptanceRateTrace)
    trace$setSynthesisOffsetTrace(synthesisOffsetTrace)
    trace$setSynthesisOffsetAcceptanceRateTrace(synthesisOffsetAcceptanceRateTrace)
    trace$setObservedSynthesisNoiseTrace(observedSynthesisNoiseTrace)
    trace$setMixtureAssignmentTrace(mixtureAssignmentTrace)
    trace$setMixtureProbabilitiesTrace(mixtureProbabilitiesTrace)
    trace$setCodonSpecificAcceptanceRateTrace(codonSpecificAcceptanceRateTrace)

    parameter$setTraceObject(trace)
    return(parameter)
  } #changed a single line

  parameter <- setBaseInfo(parameter, files)
  for (i in 1:length(files)){
    tempEnv <- new.env();
    load(file = files[i], envir = tempEnv)

    numMutationCategories <- tempEnv$paramBase$numMut
    numSelectionCategories <- tempEnv$paramBase$numSel
    max <- tempEnv$paramBase$lastIteration + 1

    if (i == 1){


      codonSpecificParameterTraceMut <- vector("list", length=numMutationCategories)
      for (j in 1:numMutationCategories) {
        codonSpecificParameterTraceMut[[j]] <- vector("list", length=length(tempEnv$mutationTrace[[j]]
        for (k in 1:length(tempEnv$mutationTrace[[j]])){
          codonSpecificParameterTraceMut[[j]][[k]] <- tempEnv$mutationTrace[[j]][[k]][1:max]
        }
      }

      codonSpecificParameterTraceSel <- vector("list", length=numSelectionCategories)
      for (j in 1:numSelectionCategories) {
        codonSpecificParameterTraceSel[[j]] <- vector("list", length=length(tempEnv$selectionTrace[[j
        for (k in 1:length(tempEnv$selectionTrace[[j]])){
          codonSpecificParameterTraceSel[[j]][[k]] <- tempEnv$selectionTrace[[j]][[k]][1:max]
        }
      }
    }else{
```

```
        curCodonSpecificParameterTraceMut <- tempEnv$mutationTrace
        curCodonSpecificParameterTraceSel <- tempEnv$selectionTrace
        combineThreeDimensionalTrace(codonSpecificParameterTraceMut, curCodonSpecificParameterTraceMut,
        combineThreeDimensionalTrace(codonSpecificParameterTraceSel, curCodonSpecificParameterTraceSel,
      }#end of if-else
    }#end of for loop (files)

    trace <- parameter$getTraceObject()

    trace$setCodonSpecificParameterTrace(codonSpecificParameterTraceMut, 0)
    trace$setCodonSpecificParameterTrace(codonSpecificParameterTraceSel, 1)

    parameter$currentMutationParameter <- tempEnv$currentMutation
    parameter$currentSelectionParameter <- tempEnv$currentSelection
    parameter$proposedMutationParameter <- tempEnv$proposedMutation
    parameter$proposedSelectionParameter <- tempEnv$proposedSelection
    parameter$setTraceObject(trace)
    return(parameter)
  }
```

Now, we load in our CDS genome file. Example uses E.coli K-12 strain .fasta file (CDS).

```
#Define .fasta file in working directory
dir <- "Escherichia_coli_str_k_12_substr_w3110_gca_000010245.ASM1024v1.cds.all.fa"

#Run function on .fasta file to import
chomp(dir)
```

```
## Now importing: Escherichia_coli_str_k_12_substr_w3110_gca_000010245.ASM1024v1.cds.all.fa
## Percentage of individual sequences w/o description: 17.32593
```

Next, we will partition the genome by sorting assigning each sequence F (forward) or R (reverse) strand. This depends on using Ensembl Bacteria's assignment (1 or -1).

Source: https://bacteria.ensembl.org/Escherichia_coli_str_k_12_substr_w3110_gca_000010245/Info/Index

```
#Take generated genome file and separate into groups based on chromosome strand assignment (1 or -1)
reverse <- genome %>% filter(str_detect(Location, ":-1$")) %>% mutate(pos = "R", assignment = 2)
forward <- genome %>% filter(str_detect(Location, ":1$")) %>% mutate(pos = "F", assignment = 1)
genome.pos <- rbind(reverse, forward)   #This changed the order, which messed up the gene.assignments!
genome.pos <- genome %>% left_join(genome.pos) %>% mutate(id = str_extract(Header, "^[^\\s]+"))
```

```
## Joining with `by = join_by(Location, Description, Type, Header, Length,
## Sequence)`
```

```
#Determine gene distribution
cat("Genes on forward strand:", nrow(forward), "-->", (nrow(forward)/nrow(genome))*100, "%",
    "\nGenes on reverse strand:", nrow(reverse), "-->", (nrow(reverse)/nrow(genome))*100, "%")
```

```
## Genes on forward strand: 2149 --> 49.71085 %
## Genes on reverse strand: 2174 --> 50.28915 %
```

We see that the split of coding sequences is roughly equal across the forward and reverse strands. Now we will begin initializing objects for the MCMC.

```
genomes <- initializeGenomeObject(
  file = "Escherichia_coli_str_k_12_substr_w3110_gca_000010245.ASM1024v1.cds.all.fa"
  )

parameter <- initializeParameterObject(
  genome = genomes,    #set genome file
  sphi = c(1, 1),    #provide phi values
  num.mixtures = 2,    #define number of groups
  gene.assignment = genome.pos$assignment, #define gene assignments
  model = "ROC"        #define model
  )

model <- initializeModelObject(parameter = parameter, model = "ROC")

mcmc <- initializeMCMCObject(samples = 5000, thinning = 10, adaptive.width=50)

#Set restart settings
setRestartSettings(mcmc = mcmc, filename = "restart_out_Ecoli_split_sharedSelection_sphi1_09302024",
                   samples = 2500)
```

Run the model [Current run time ~1.5hrs]

```
#runMCMC(mcmc = mcmc, genome = genomes, model = model)
```

Make sure to save your files immediately after the run.

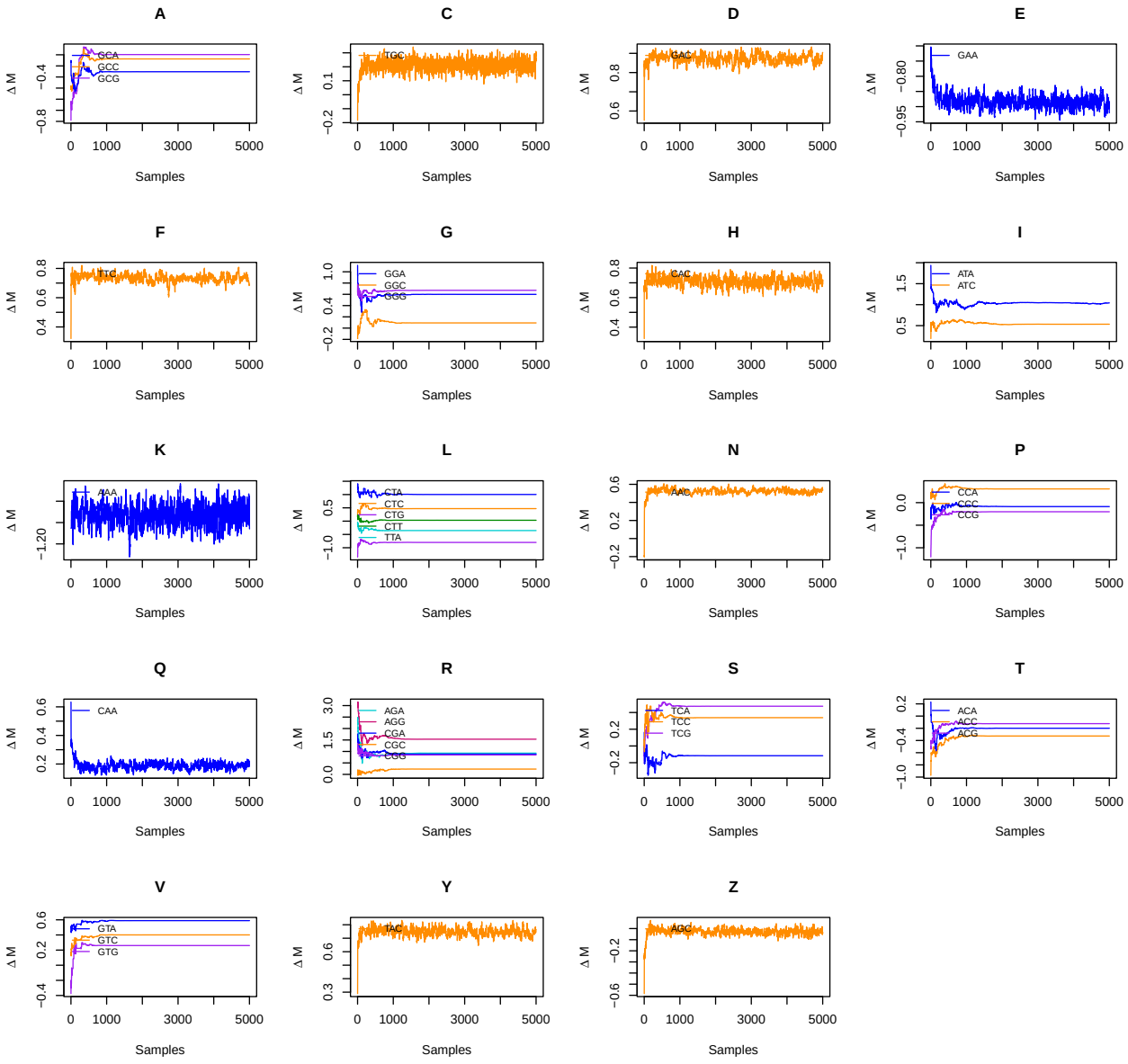
```
#writeParameterObject(parameter = parameter, file = "parameter.file.name.Rda")
#writeMCMCObject(mcmc = mcmc, file = "MCMC.file.name.Rda")
```

Use the following code to load in a previous run.

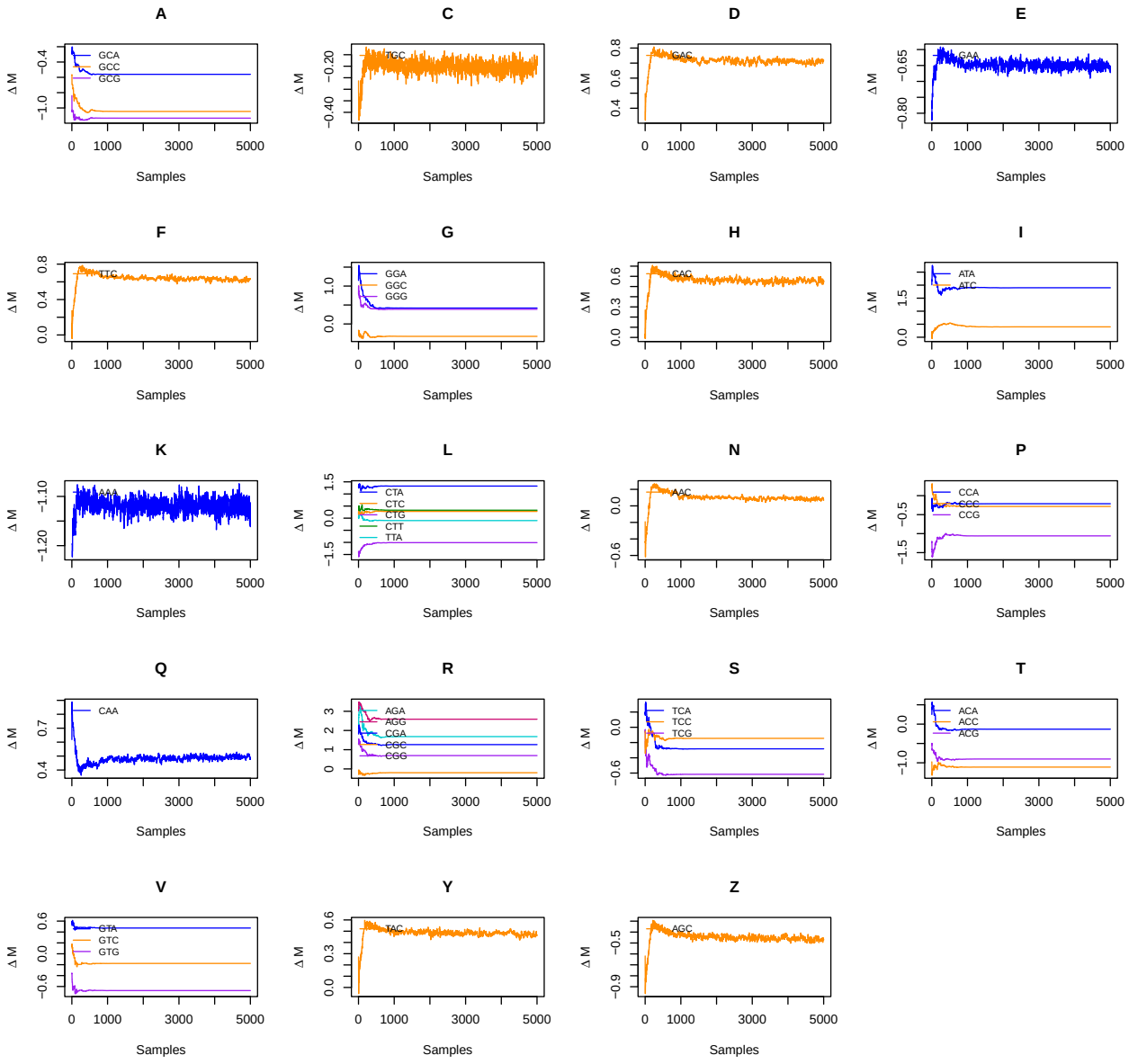
```
mcmc <- loadMCMCObject(file = "mcmc_out_Ecoli_split_sharedSelection_sphiObsMeanPhi_10012024.Rda")

files <- "parameter_out_Ecoli_split_sharedSelection_sphiObsMeanPhi_10012024.Rda"

parameter <- loadROCParameterObject(parameter, files)
```

Now that we have our model run, let's see if we've reached convergence. The graphs are large, so you might get some errors about dimensions.

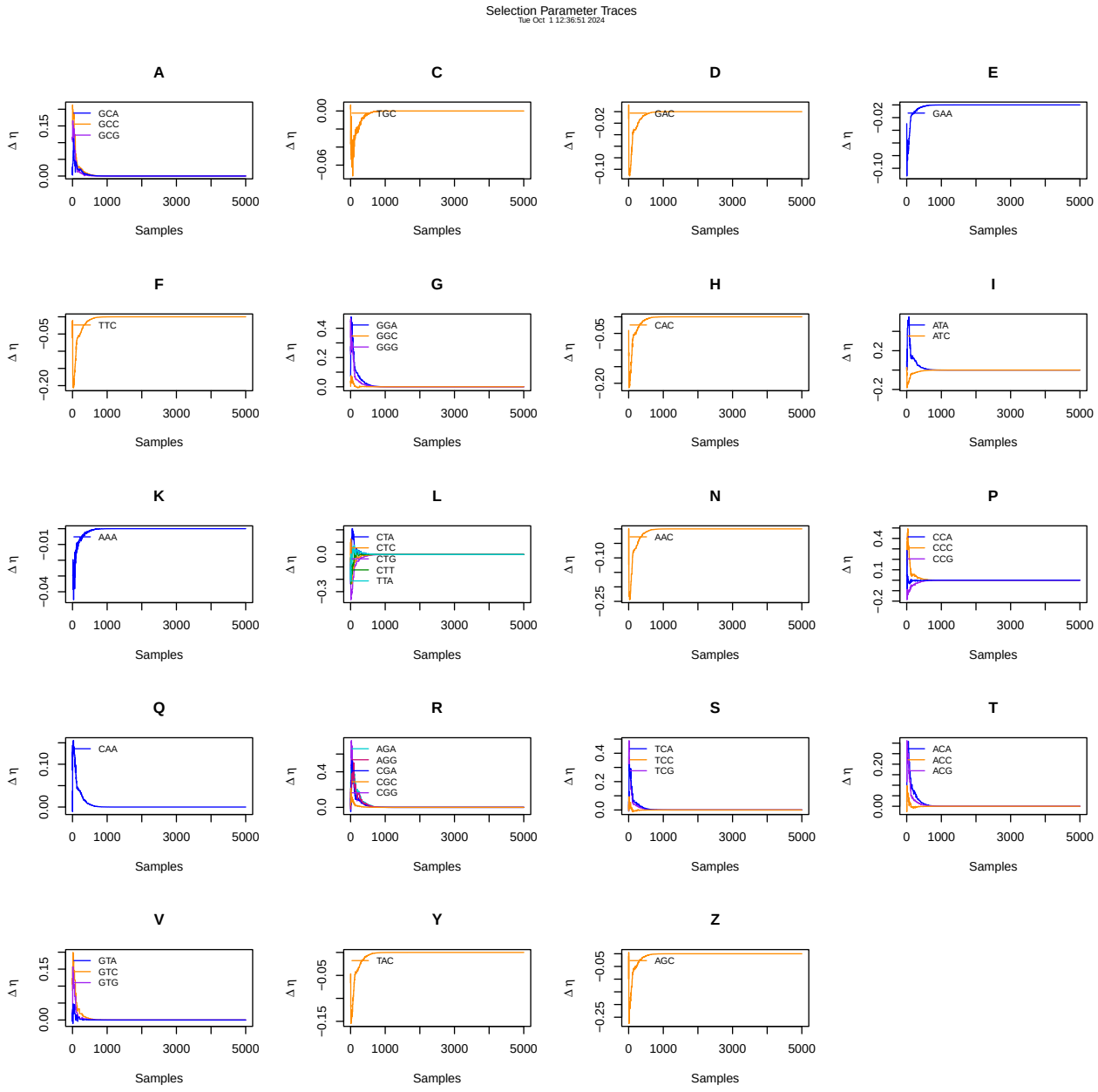
```
trace <- getTrace(parameter)  #store the trace (i.e. record of all iterations)

#Visualize the mutation parameters
#We are looking for the lines to level out.
plot.diag.1m <- plot(x = trace, what = "Mutation", mixture = 1)
```




```
plot.diag.2m <- plot(x = trace, what = "Mutation", mixture = 2)
```



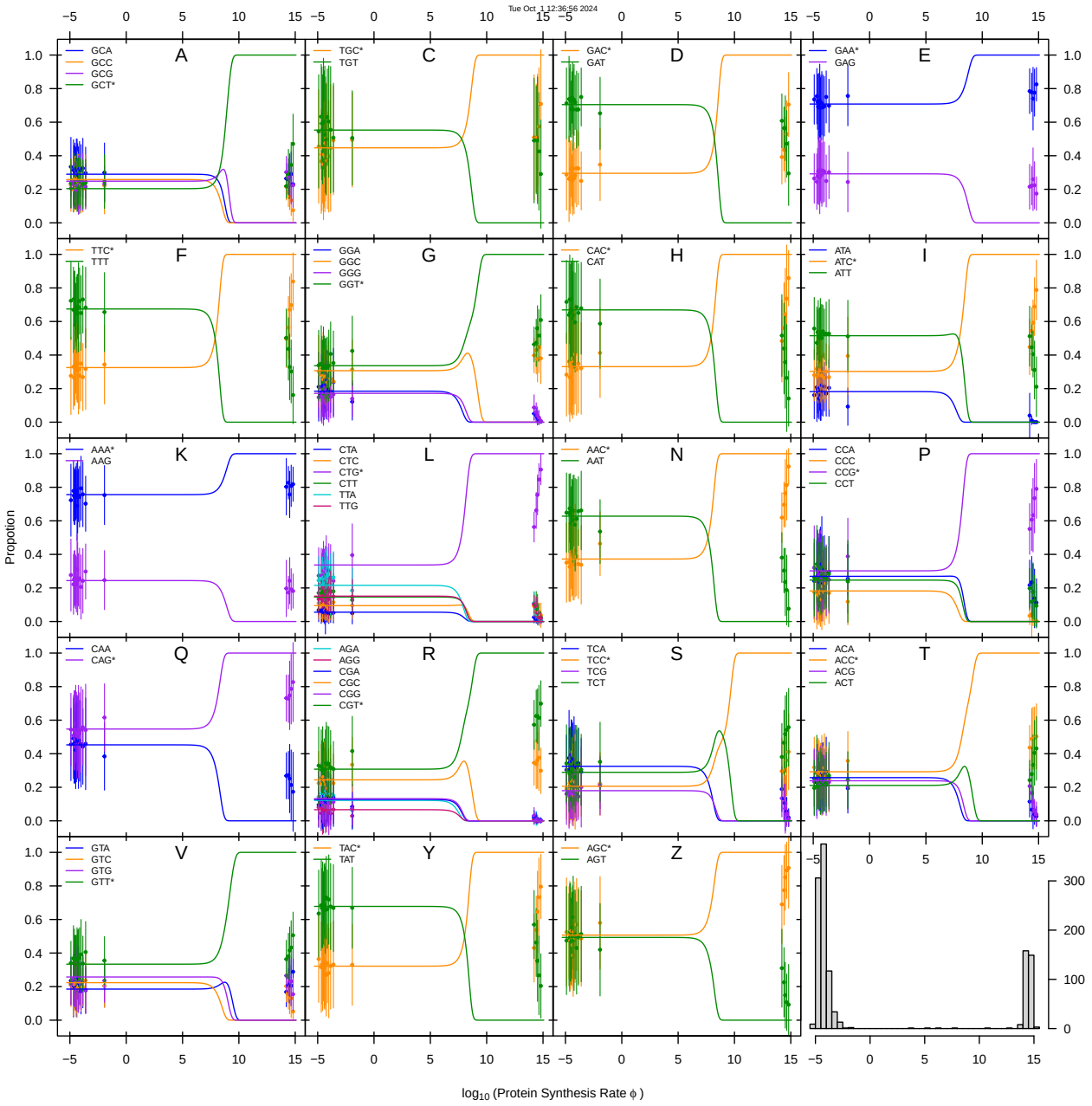

```
#We can also visualize the selection parameter
plot.diag.1s <- plot(x = trace, what = "Selection", mixture = 1)
```



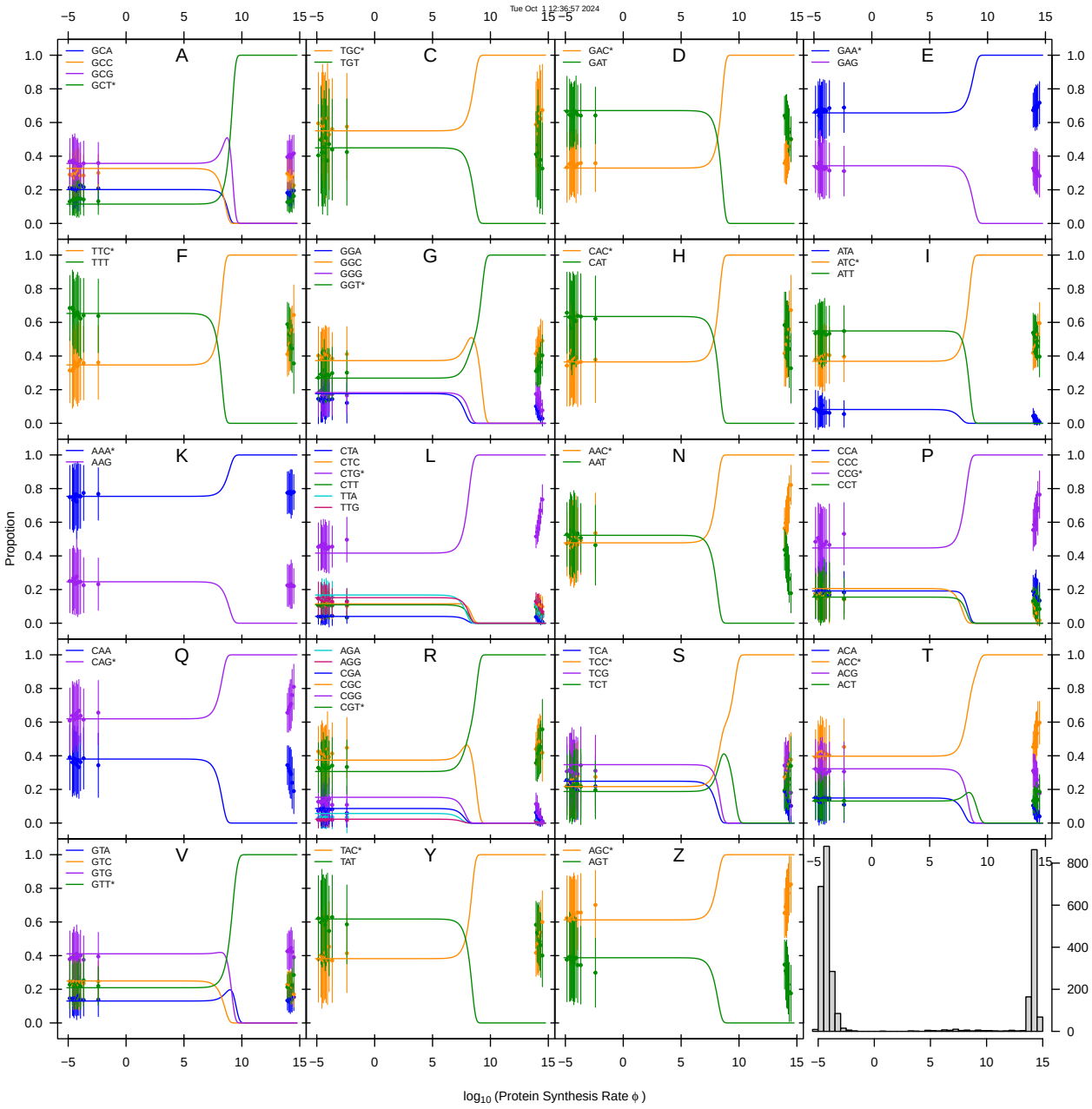

```
plot.diag.2s <- plot(x = trace, what = "Selection", mixture = 2)
```

The plots above should show each trace becoming more and more level. If not, increase the number of smaples to run during the MCMC. Now we can visualize codon use proportion as it relates to protein synthesis rate phi. We can also visualize how selection and mutation parameters correlated between our mixtures.

```
#visualize the results of the model fit
plot(x = model, genome = genomes, samples = 3000, mixture = 1)
```

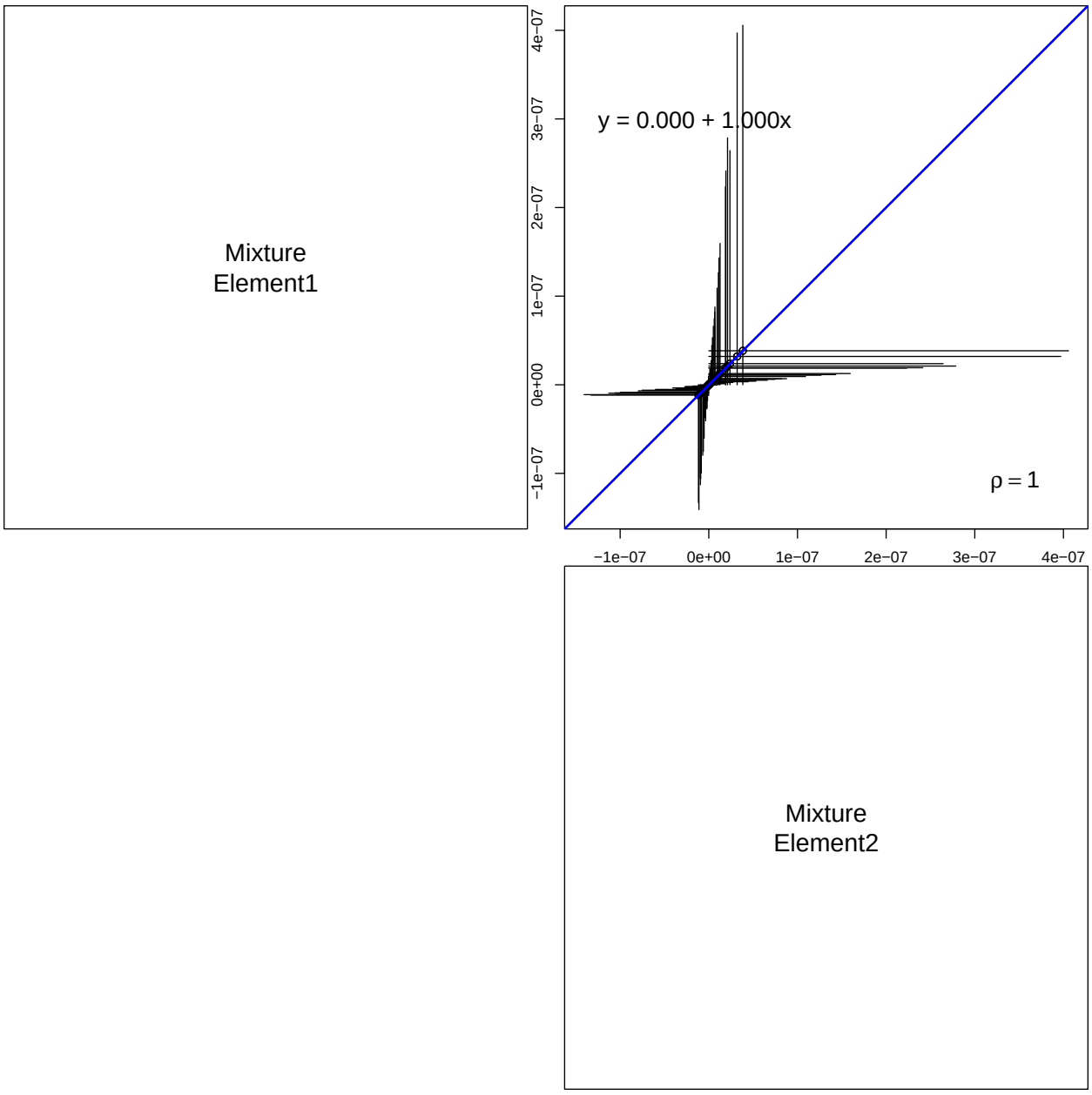



```
plot(x = model, genome = genomes, samples = 3000, mixture = 2)
```

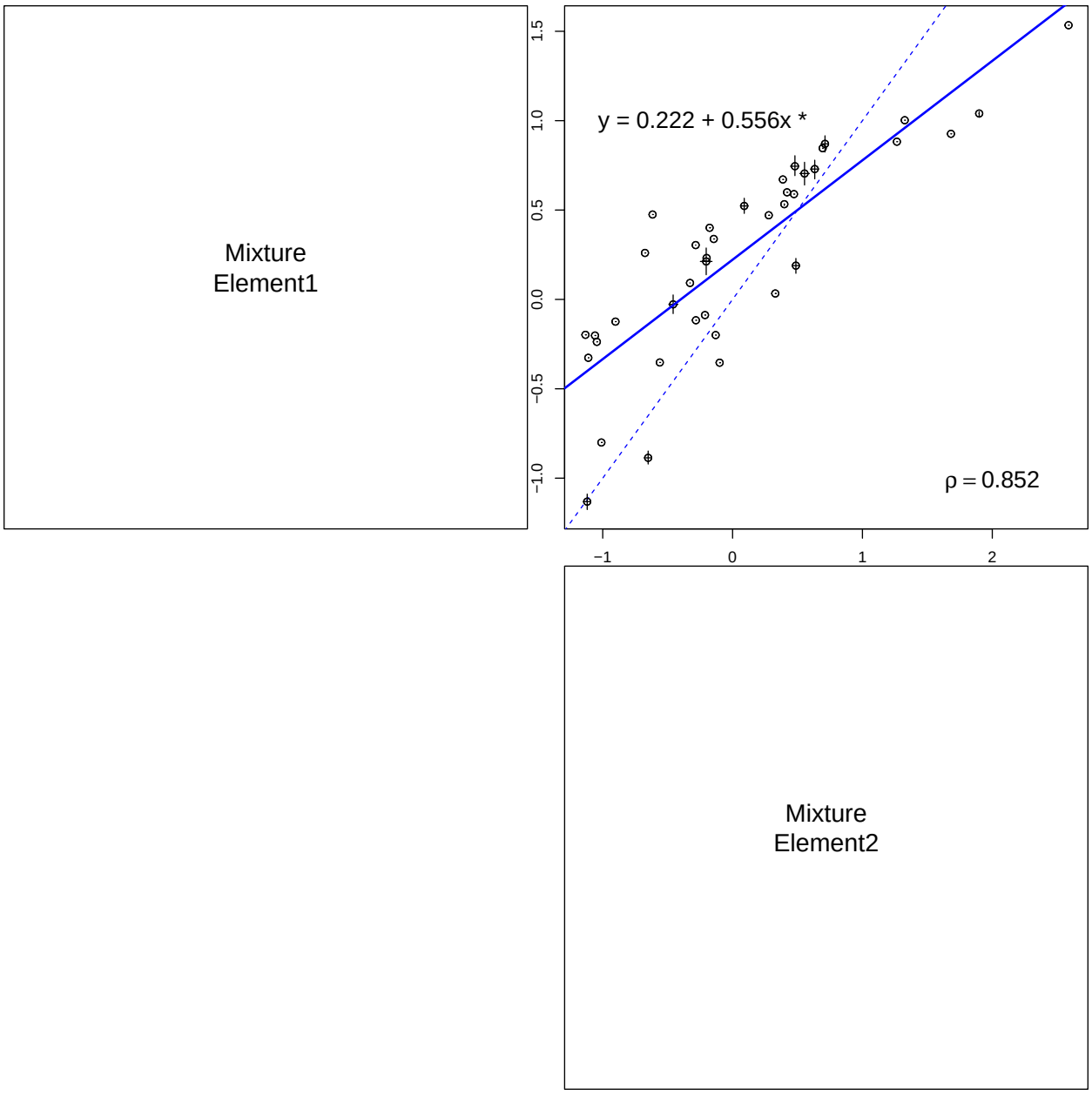



```
#This will compare different 'mixtures' i.e. gene sets
plot(parameter, what = "Selection", samples = 3000)
```

```
## Warning in summary.lm(lm.line): essentially perfect fit: summary may be
## unreliable
## Warning in summary.lm(lm.line): essentially perfect fit: summary may be
## unreliable
## Warning in summary.lm(lm.line): essentially perfect fit: summary may be
## unreliable
## Warning in summary.lm(lm.line): essentially perfect fit: summary may be
## unreliable
```



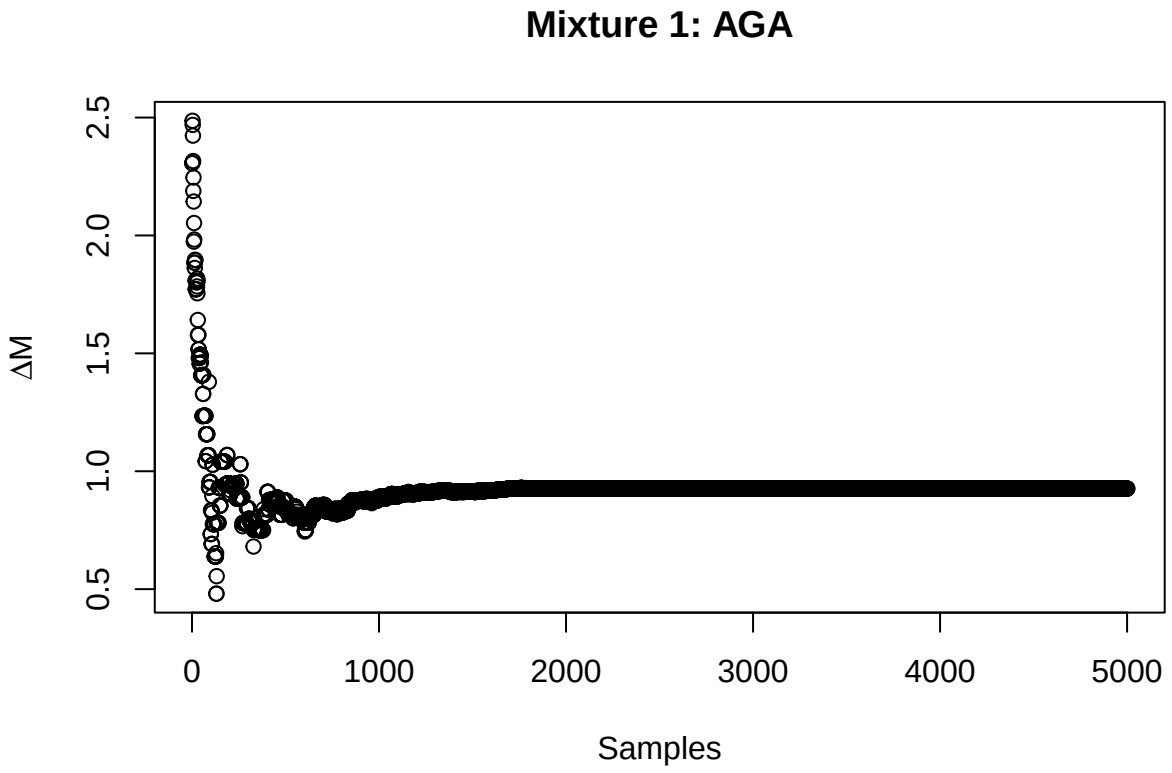

```
plot(parameter, what = "Mutation", samples = 3000)
```



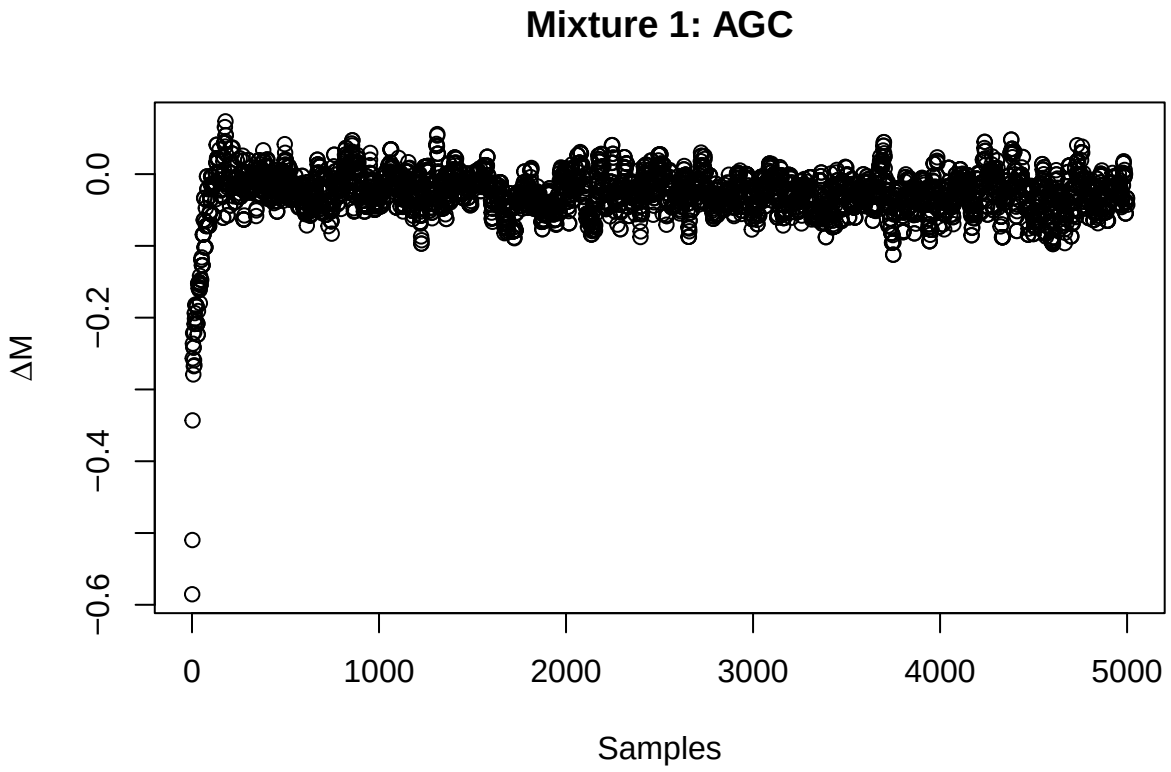
Now, we want to extract the mutation trace for each individual codon.

```
#Now to grab codon specific parameters
mutationTrace <- trace$getCodonSpecificParameterTrace(0)

#Take object of traces and feed into bayesplot
names(mutationTrace) <- c("mix1", "mix2")

#Get list of codons
names.aa <- aminoAcids()
names.aa <- setdiff(names.aa, c("W", "X", "M")) #remove W, X, and M

#This should be the order conserved in mutationTrace[[i]]
AA.df <- names.aa %>%                  #pipe amino acids
```

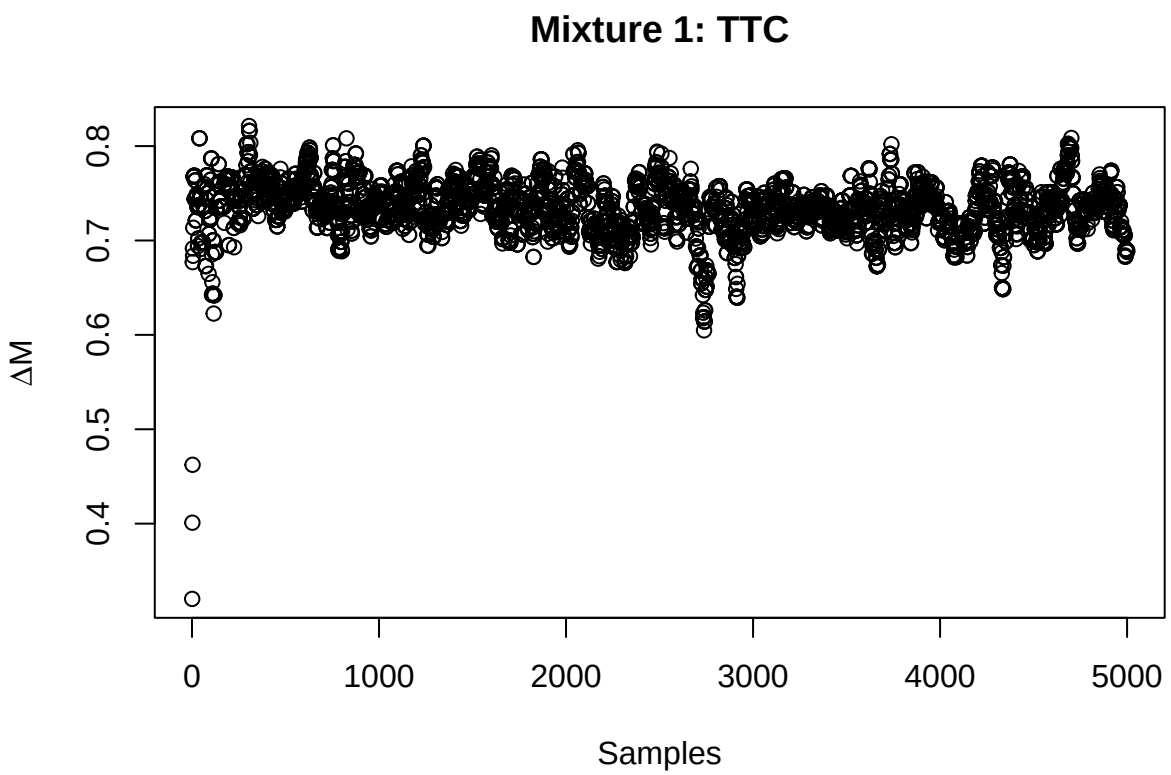
```
  data.frame(aa = names.aa) %>%        #make a dataframe
  select(-.) %>%                       #remove duplicated column
  mutate(Codon = lapply(aa, AAToCodon)) %>% #make new column with associated codons for each amino acid
  mutate(Codon = map(Codon, ~ .x[-length(.x)])) %>% #remove the reference codon (last alphabetical codo
  unnest(cols = c(Codon))                          #unnest column data

#This assumes that the codons are sorted by amino acid, then codon sequence (i.e. A: GCA, GCC, GCG, etc
names(mutationTrace[[1]]) <- AA.df$Codon
names(mutationTrace[[2]]) <- AA.df$Codon

#To verify this data matches our plotted trace data, plot a few codons individually to compare
plot(mutationTrace$mix1$AGA, main = "Mixture 1: AGA", ylab = expression(Delta * M), xlab = "Samples")
```



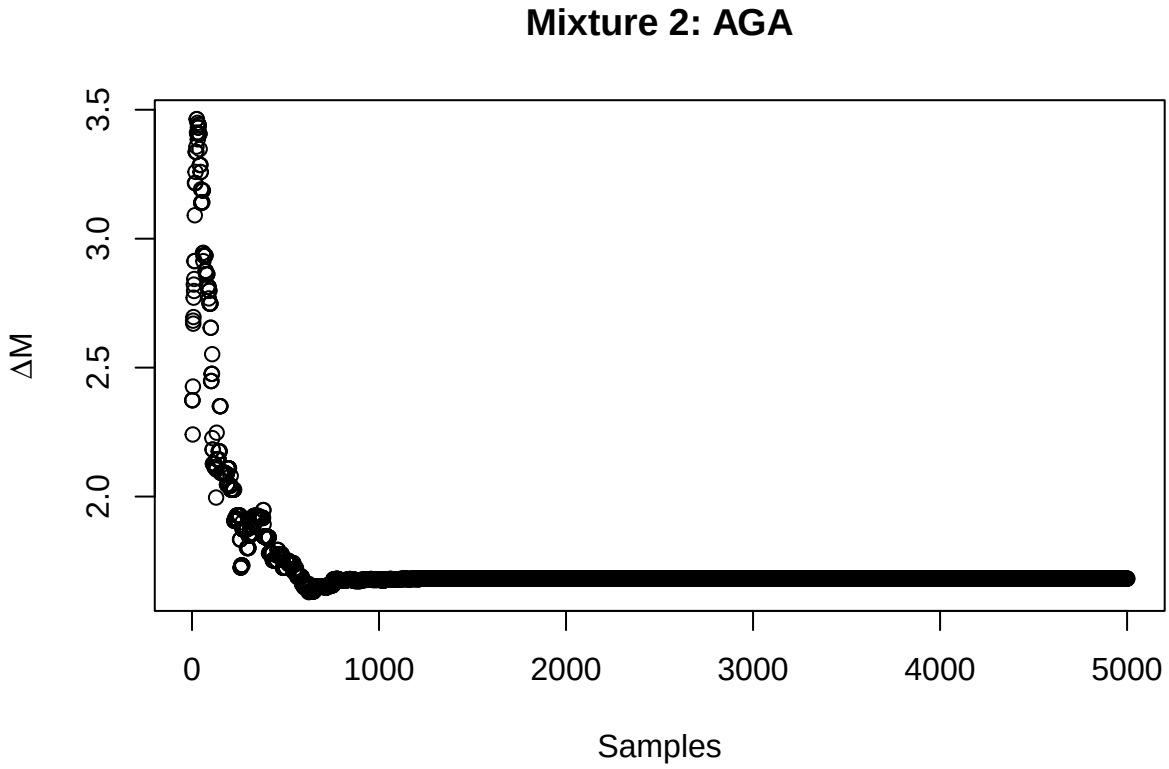

```
plot(mutationTrace$mix1$AGC, main = "Mixture 1: AGC", ylab = expression(Delta * M), xlab = "Samples")
```



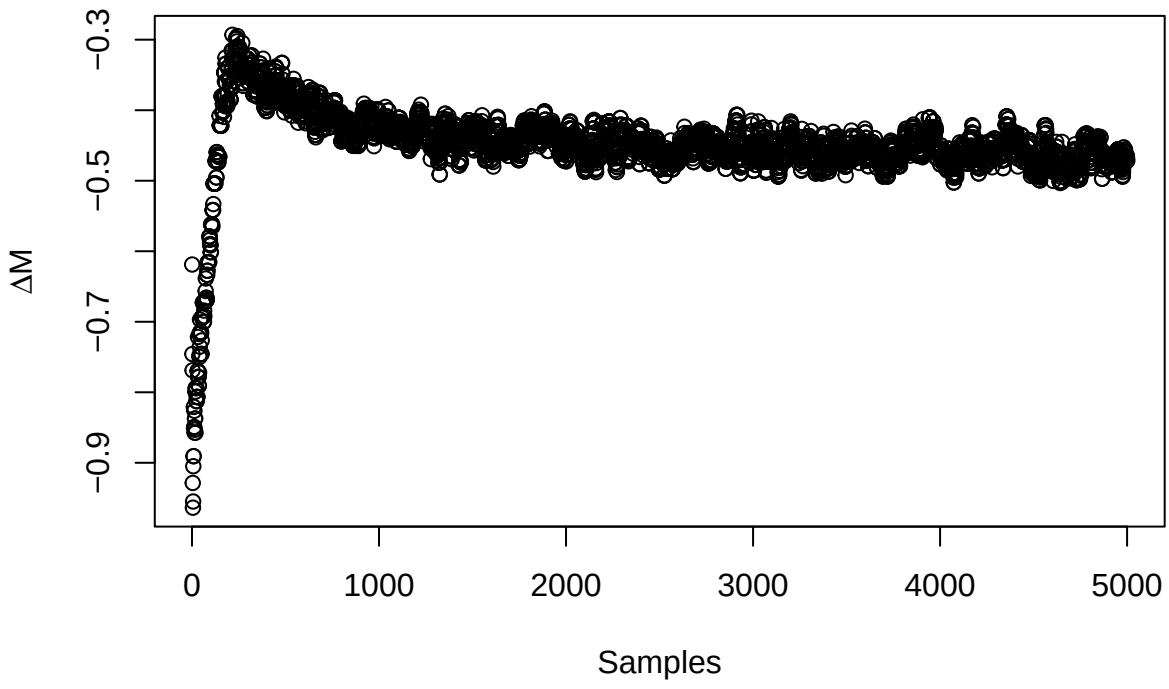

```
plot(mutationTrace$mix1$TTC, main = "Mixture 1: TTC", ylab = expression(Delta * M), xlab = "Samples")
```



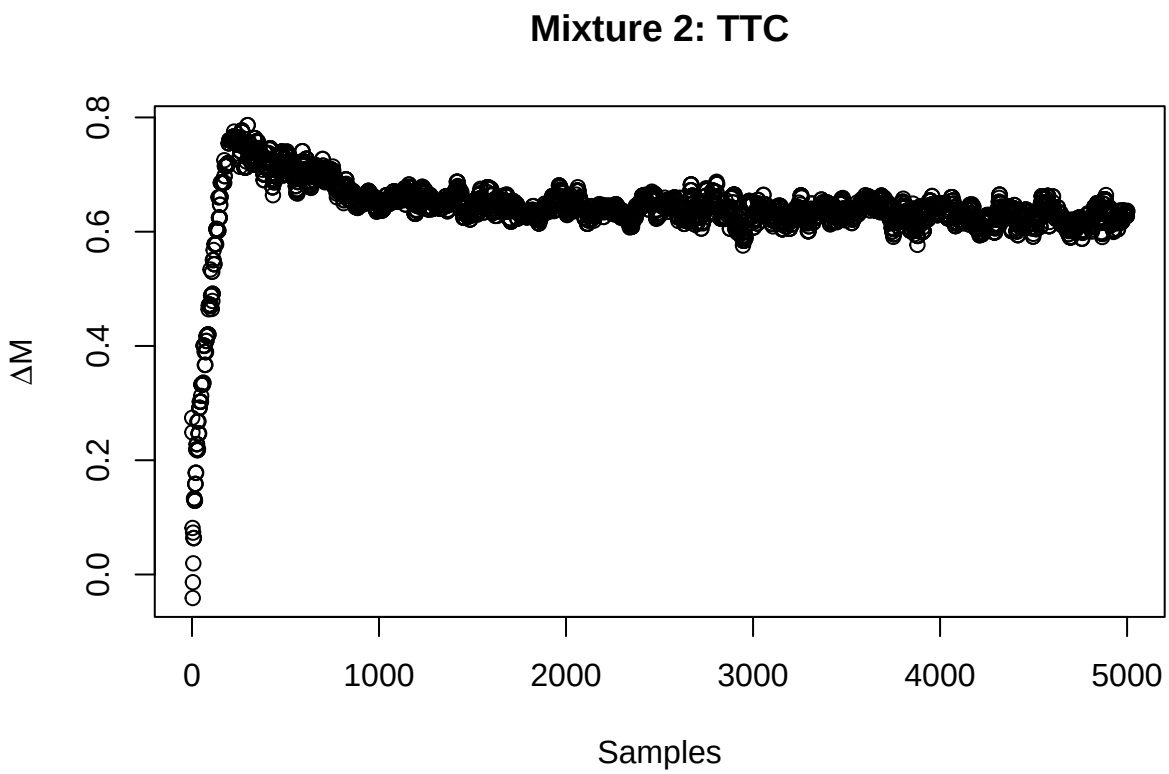
```
plot(mutationTrace$mix2$AGA, main = "Mixture 2: AGA", ylab = expression(Delta * M), xlab = "Samples")
```




```
plot(mutationTrace$mix2$AGC, main = "Mixture 2: AGC", ylab = expression(Delta * M), xlab = "Samples")
```

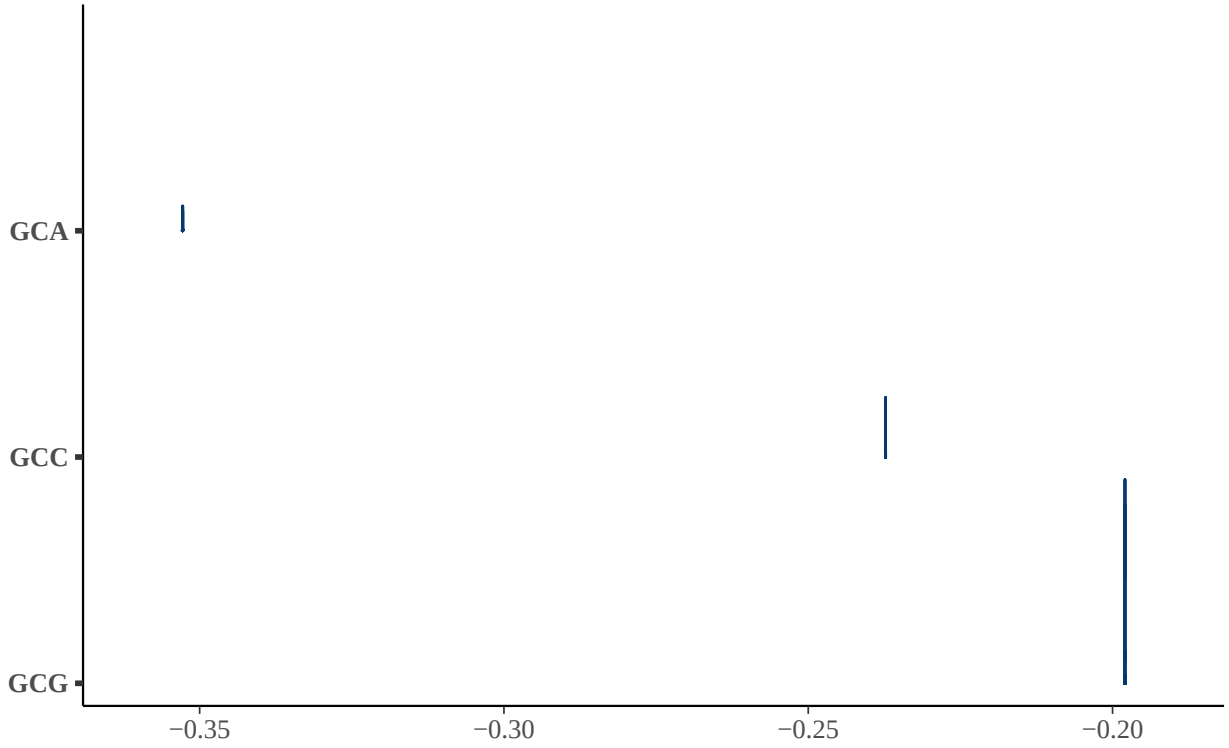


```r
plot(mutationTrace$mix2$TTC, main = "Mixture 2: TTC", ylab = expression(Delta * M), xlab = "Samples")
```

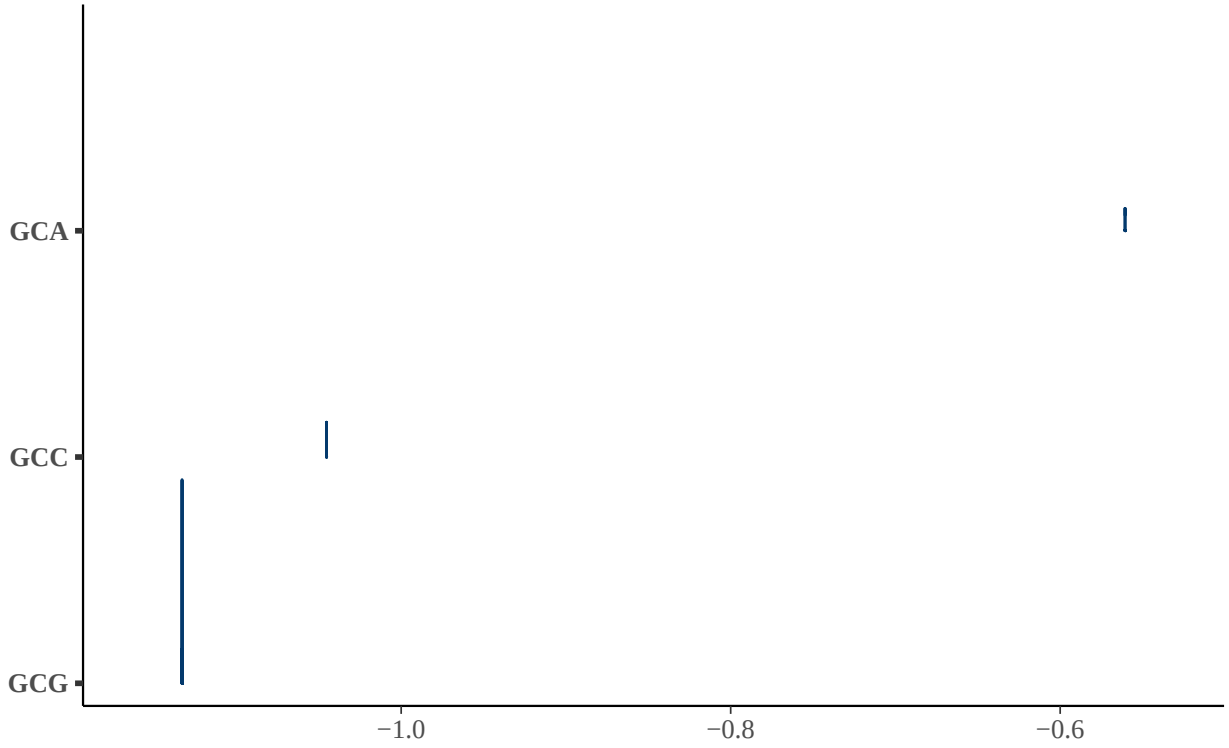


Now that we have the posterior distributions of our estimated parameter, let's try to plot it using Bayesplot.

```r
#Making trace data frames and discard burn-in
mutTrace.1 <- mutationTrace$mix1 %>% data.frame() %>% slice(-1:-2000)
mutTrace.2 <- mutationTrace$mix2 %>% data.frame() %>% slice(-1:-2000)

#Moving on to Bayesplot
posterior.1 <- as.matrix(mutTrace.1)                    #fit trace into posterior matrix
posterior.2 <- as.matrix(mutTrace.2)

mcmc_areas(posterior.1,                                 #supply the posterior matrix
           pars = c("GCA", "GCC", "GCG"),               #supply the desired traces to plot
           prob = 0.8) +                                #supply desired interval
  ggtitle("Posterior distributions: Codons of Alanine [Mixture 1]",  #supply title
          "with medians and 80% intervals")# +
```



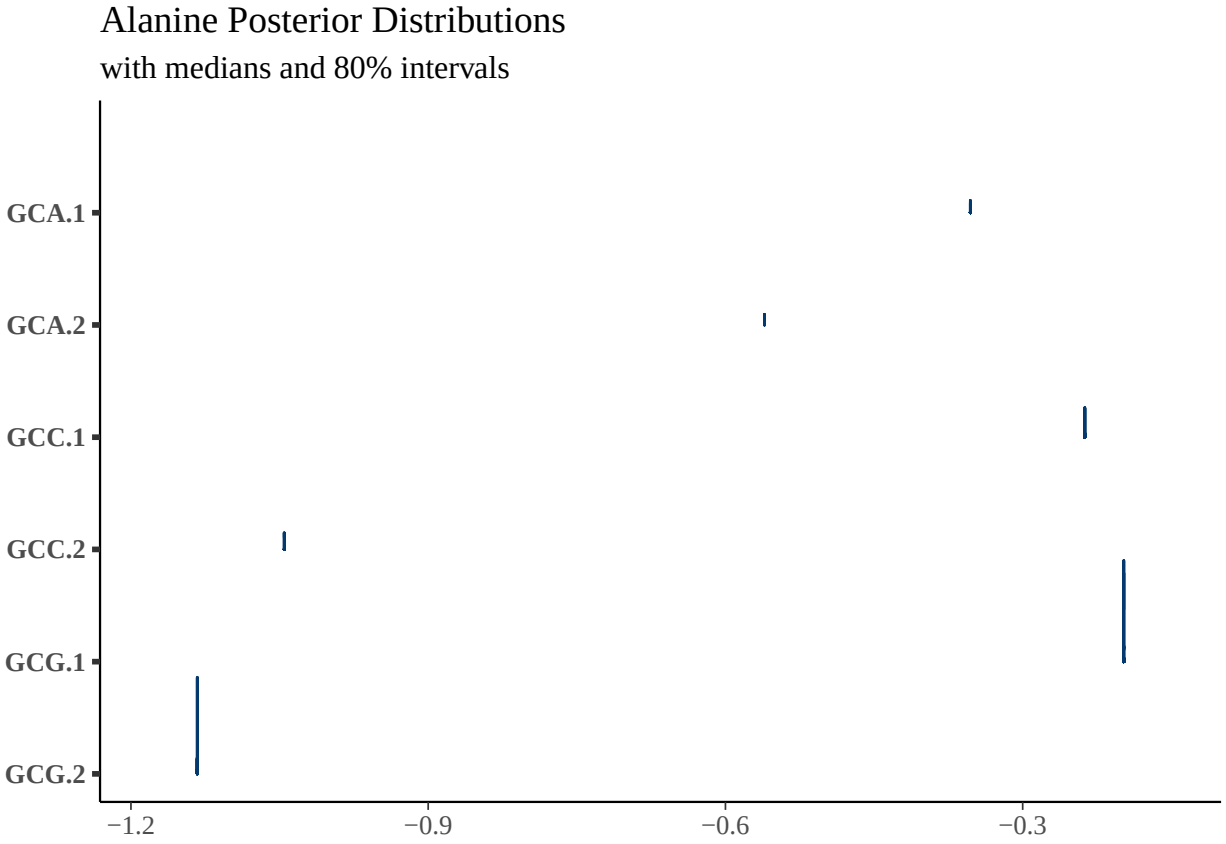

```
  #coord_cartesian(xlim = c(-1.5, 0.5))                  #adjust x-axis

mcmc_areas(posterior.2,                                  #supply the posterior matrix
           pars = c("GCA", "GCC", "GCG"),                #supply the desired traces to plot
           prob = 0.8) +                                 #supply desired interval
  ggtitle("Posterior distributions: Codons of Alanine [Mixture 2]",  #supply title
          "with medians and 80% intervals")# +
```




```
  #coord_cartesian(xlim = c(-1.5, 0.5))                    #adjust x-axis
```

We can also directly compare codons across mixtures

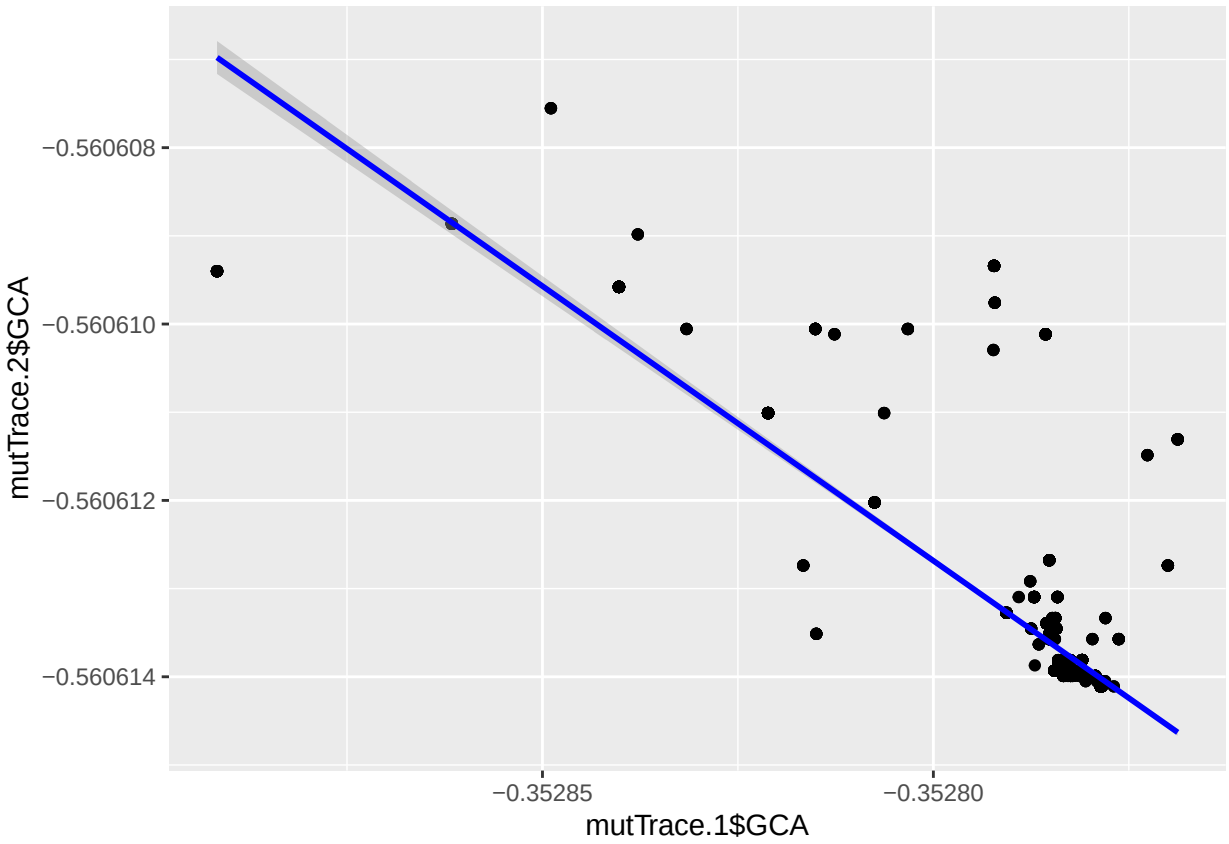
```
#Example: show all codons of alanine across the mixtures
alanine <- data.frame(
  GCA.1 = mutTrace.1$GCA,
  GCA.2 = mutTrace.2$GCA,
  GCC.1 = mutTrace.1$GCC,
  GCC.2 = mutTrace.2$GCC,
  GCG.1 = mutTrace.1$GCG,
  GCG.2 = mutTrace.2$GCG
)

posterior <- as.matrix(alanine)

mcmc_areas(posterior, prob = 0.8) +
  ggtitle("Alanine Posterior Distributions","with medians and 80% intervals")
```



Now, to calculate some correlations between codons on different strands.

```
#Testing a visualization method
ggplot() +
  geom_point(aes(x = mutTrace.1$GCA, y = mutTrace.2$GCA)) +
  geom_smooth(aes(x = mutTrace.1$GCA, y = mutTrace.2$GCA), method = "lm", color = "blue", se = TRUE) # 
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

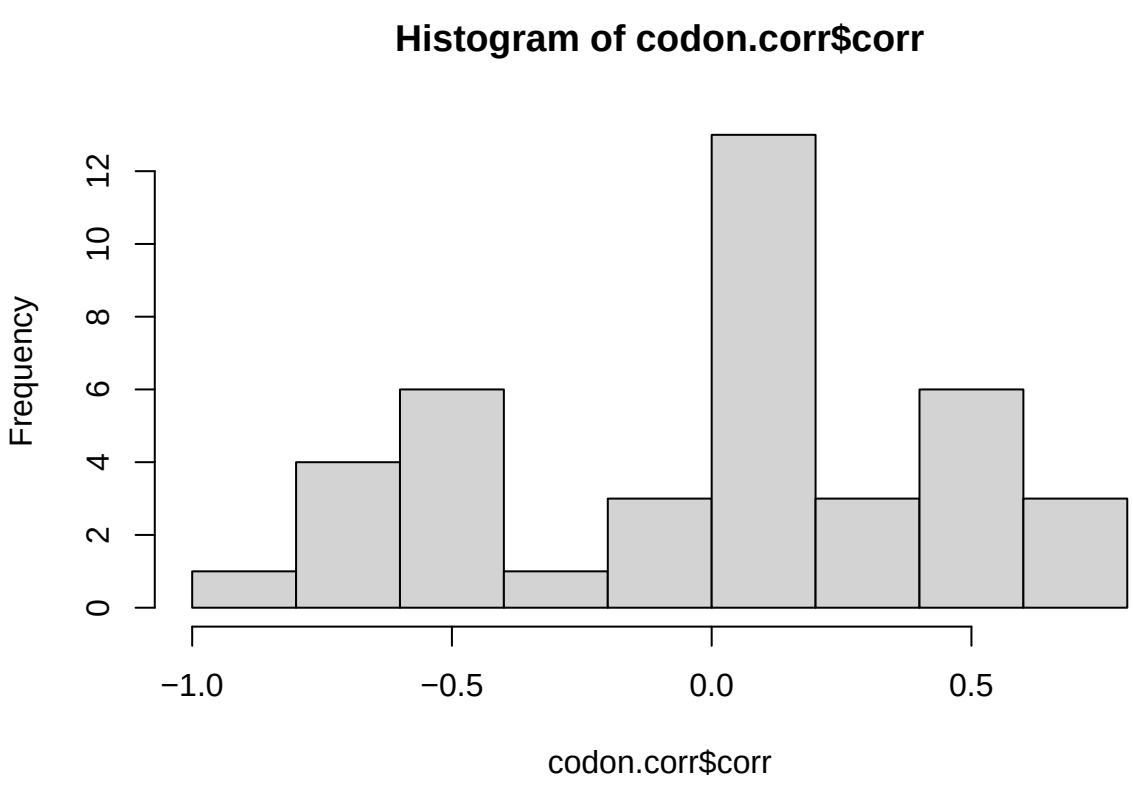



```
regression <- lm(mutTrace.2$GCA ~ mutTrace.1$GCA)
summary(regression)
```

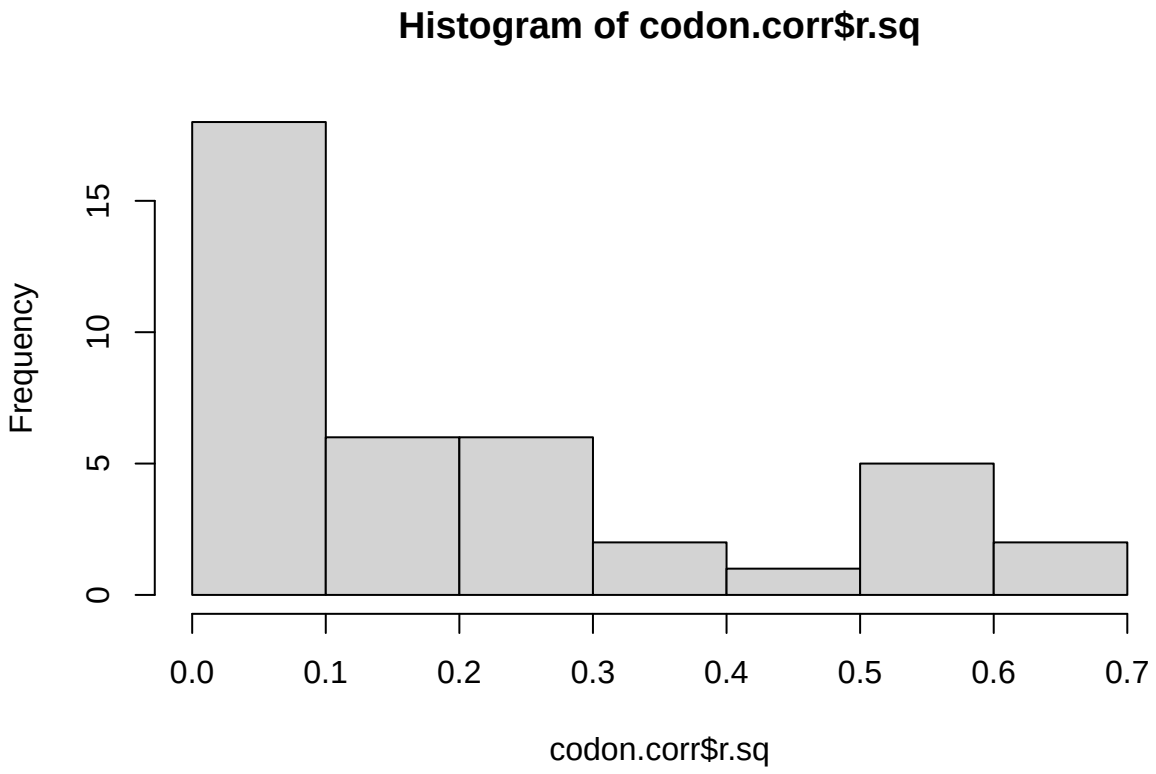
```
##
## Call:
## lm(formula = mutTrace.2$GCA ~ mutTrace.1$GCA)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.423e-06 -1.853e-07 -1.815e-07 -9.310e-08  3.827e-06
##
## Coefficients:
##                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)    -0.5825863  0.0003161 -1843.27   <2e-16 ***
## mutTrace.1$GCA -0.0622834  0.0008959   -69.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.821e-07 on 2999 degrees of freedom
## Multiple R-squared:  0.6171, Adjusted R-squared:  0.617
## F-statistic:  4833 on 1 and 2999 DF,  p-value: < 2.2e-16
```

```
#Calculate the correlation between each 'pair' of codons (mixture1:mixture2)
correlations <- sapply(names(mutTrace.1), function(col) {
  cor(mutTrace.1[[col]], mutTrace.2[[col]])
})
```

```r
codon.corr <- data.frame(
  codon = names(correlations),
  corr = correlations,
  r.sq = correlations^2
)

hist(codon.corr$corr)
```




```r
hist(codon.corr$r.sq)
```




```
print(codon.corr)
```

```
##     codon        corr         r.sq
## GCA   GCA -0.78555018 0.6170890868
## GCC   GCC -0.45078992 0.2032115508
## GCG   GCG -0.30544312 0.0932955014
## TGC   TGC  0.09582656 0.0091827291
## GAC   GAC  0.12522175 0.0156804855
## GAA   GAA  0.03892324 0.0015150184
## TTC   TTC -0.03573077 0.0012766880
## GGA   GGA  0.03587699 0.0012871585
## GGC   GGC -0.77051225 0.5936891203
## GGG   GGG  0.09692757 0.0093949533
## CAC   CAC -0.04340362 0.0018838743
## ATA   ATA  0.19369255 0.0375168053
## ATC   ATC  0.76658832 0.5876576580
## AAA   AAA  0.12797800 0.0163783682
## CTA   CTA -0.74401390 0.5535566768
## CTC   CTC -0.40815015 0.1665865426
## CTG   CTG  0.72370858 0.5237541031
## CTT   CTT  0.13792869 0.0190243237
## TTA   TTA  0.07427618 0.0055169513
## AAC   AAC  0.23741445 0.0563656234
## CCA   CCA -0.42484481 0.1804931102
## CCC   CCC -0.72912297 0.5316203078
## CCG   CCG  0.40802534 0.1664846745
## CAA   CAA  0.01486856 0.0002210740
## AGA   AGA  0.37797231 0.1428630681
## AGG   AGG -0.10179900 0.0103630373
```

```
## CGA   CGA  0.06289363 0.0039556086
## CGC   CGC  0.40786470 0.1663536118
## CGG   CGG  0.38670256 0.1495388711
## TCA   TCA  0.47474693 0.2253846465
## TCC   TCC  0.48258048 0.2328839189
## TCG   TCG  0.69517522 0.4832685927
## ACA   ACA -0.47950791 0.2299278332
## ACC   ACC  0.48890181 0.2390249751
## ACG   ACG -0.59381767 0.3526194289
## GTA   GTA  0.56979174 0.3246626264
## GTC   GTC -0.51236998 0.2625229948
## GTG   GTG -0.83291858 0.6937533573
## TAC   TAC  0.09449087 0.0089285242
## AGC   AGC  0.01601181 0.0002563781
```

```r
# Run regression analysis and store results
regression_results <- sapply(names(mutTrace.1), function(col) {
  model <- lm(mutTrace.2[[col]] ~ mutTrace.1[[col]])
  summary(model)$coefficients[2, ]  # Get the coefficient and its statistics
})

# Create a data frame from the results
regression_df <- data.frame(
  Codon = names(mutTrace.1),
  Estimate = regression_results[1, ],
  Std.Error = regression_results[2, ],
  t.value = regression_results[3, ],
  p.value = regression_results[4, ]
)

print(regression_df)
```

```
##     Codon     Estimate    Std.Error     t.value       p.value
## GCA   GCA -0.062283448 0.0008958996 -69.5205651  0.000000e+00
## GCC   GCC -0.178014548 0.0064367194 -27.6560990 3.738899e-150
## GCG   GCG -0.030696048 0.0017474177 -17.5665199  7.888671e-66
## TGC   TGC  0.058062249 0.0110132759   5.2720235  1.445017e-07
## GAC   GAC  0.073278023 0.0106016584   6.9119396  5.816454e-12
## GAA   GAA  0.023167245 0.0108604536   2.1331747  3.299136e-02
## TTC   TTC -0.023520858 0.0120128246  -1.9579790  5.032507e-02
## GGA   GGA  0.001373780 0.0006987683   1.9660019  4.939051e-02
## GGC   GGC -0.651817764 0.0098466305 -66.1970372  0.000000e+00
## GGG   GGG  0.004169306 0.0007817691   5.3331682  1.036848e-07
## CAC   CAC -0.025386368 0.0106703071  -2.3791601  1.741436e-02
## ATA   ATA  0.017903122 0.0016558622  10.8119637  9.347652e-27
## ATC   ATC  0.137999186 0.0021108388  65.3764699  0.000000e+00
## AAA   AAA  0.076091804 0.0107678344   7.0665838  1.966431e-12
## CTA   CTA -0.016130307 0.0002645188 -60.9798166  0.000000e+00
## CTC   CTC -0.041527010 0.0016961050 -24.4837501 7.624417e-121
## CTG   CTG  0.199559534 0.0034748507  57.4296715  0.000000e+00
## CTT   CTT  0.015285348 0.0020042947   7.6262973  3.221989e-14
## TTA   TTA  0.006781304 0.0016625477   4.0788630  4.643462e-05
## AAC   AAC  0.162888447 0.0121701735  13.3842338  1.008666e-39
## CCA   CCA -0.017023292 0.0006623710 -25.7005396 8.066185e-132
```

```
## CCC   CCC -0.042561951 0.0007295105 -58.3431651  0.000000e+00
## CCG   CCG  0.055023178 0.0022481594  24.4747672 9.160819e-121
## CAA   CAA  0.009299114 0.0114192223   0.8143386  4.155156e-01
## AGA   AGA  0.043517864 0.0019464554  22.3574936 1.562713e-102
## AGG   AGG -0.021859801 0.0039007883  -5.6039445  2.285242e-08
## CGA   CGA  0.002450318 0.0007100140   3.4510841  5.660564e-04
## CGC   CGC  0.023022851 0.0009411215  24.4632083 1.160101e-120
## CGG   CGG  0.075948344 0.0033073508  22.9634982 1.236218e-107
## TCA   TCA  0.061817594 0.0020926893  29.5397861 1.482742e-168
## TCC   TCC  0.071786596 0.0023791176  30.1736216 6.738612e-175
## TCG   TCG  0.175321667 0.0033104383  52.9602576  0.000000e+00
## ACA   ACA -0.092010916 0.0030748297 -29.9239068 2.167758e-172
## ACC   ACC  0.077216290 0.0025158472  30.6919634 3.878423e-180
## ACG   ACG -0.018629352 0.0004609313 -40.4167624 1.672668e-285
## GTA   GTA  0.037468630 0.0009867879  37.9702966 5.940391e-258
## GTC   GTC -0.050557444 0.0015473460 -32.6736513 1.387633e-200
## GTG   GTG -0.178654344 0.0021674968 -82.4242713  0.000000e+00
## TAC   TAC  0.057184350 0.0110014924   5.1978721  2.151011e-07
## AGC   AGC  0.011436177 0.0130405664   0.8769693  3.805735e-01
```