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.

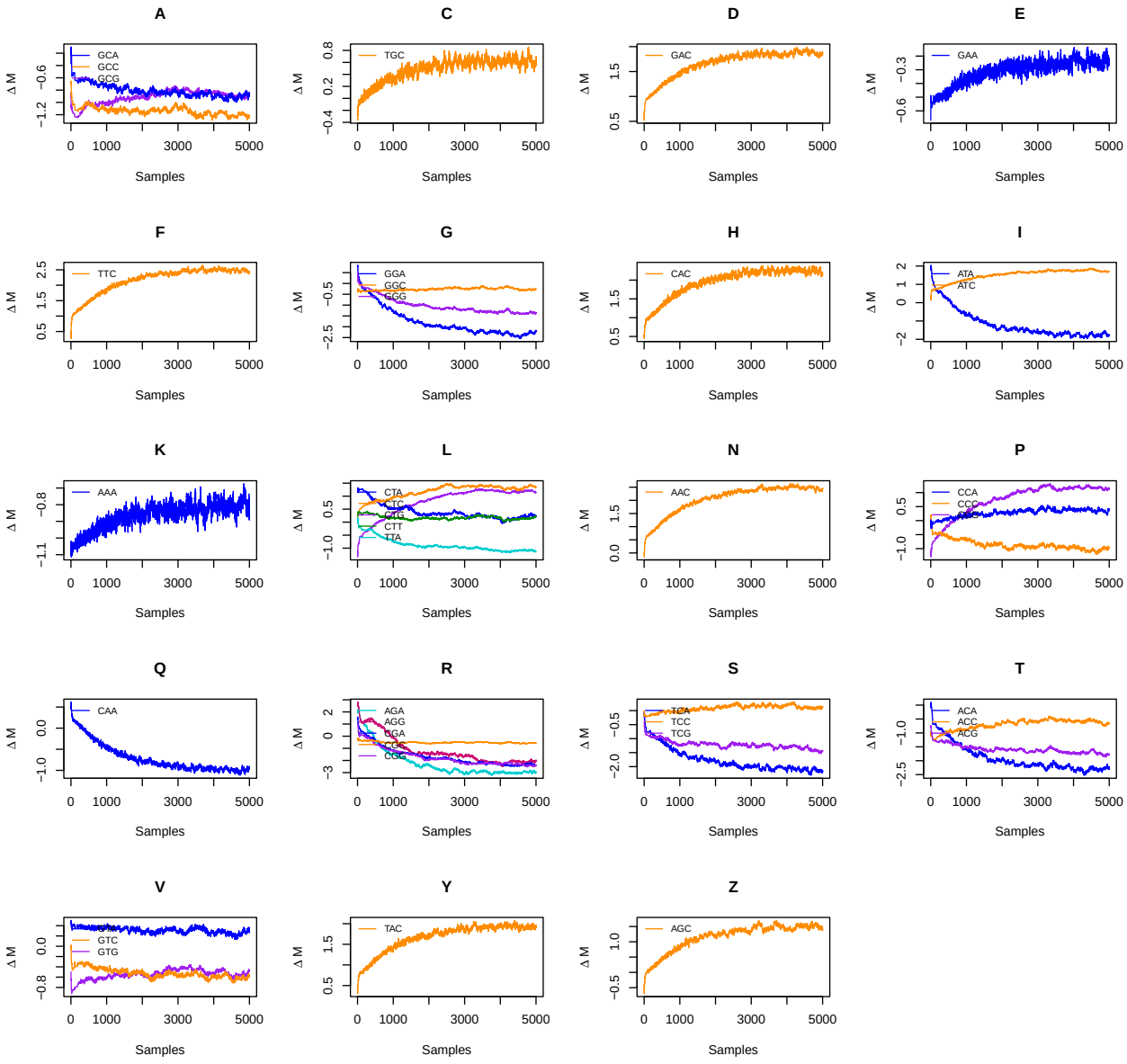
```
#writeParameterObject(parameter = parameter, file = "parameter.file.name.Rda")
#writeMCMCObject(mcmc = mcmc, file = "MCMC.file.name.Rda")
```

Use the following code to load in a previous run.

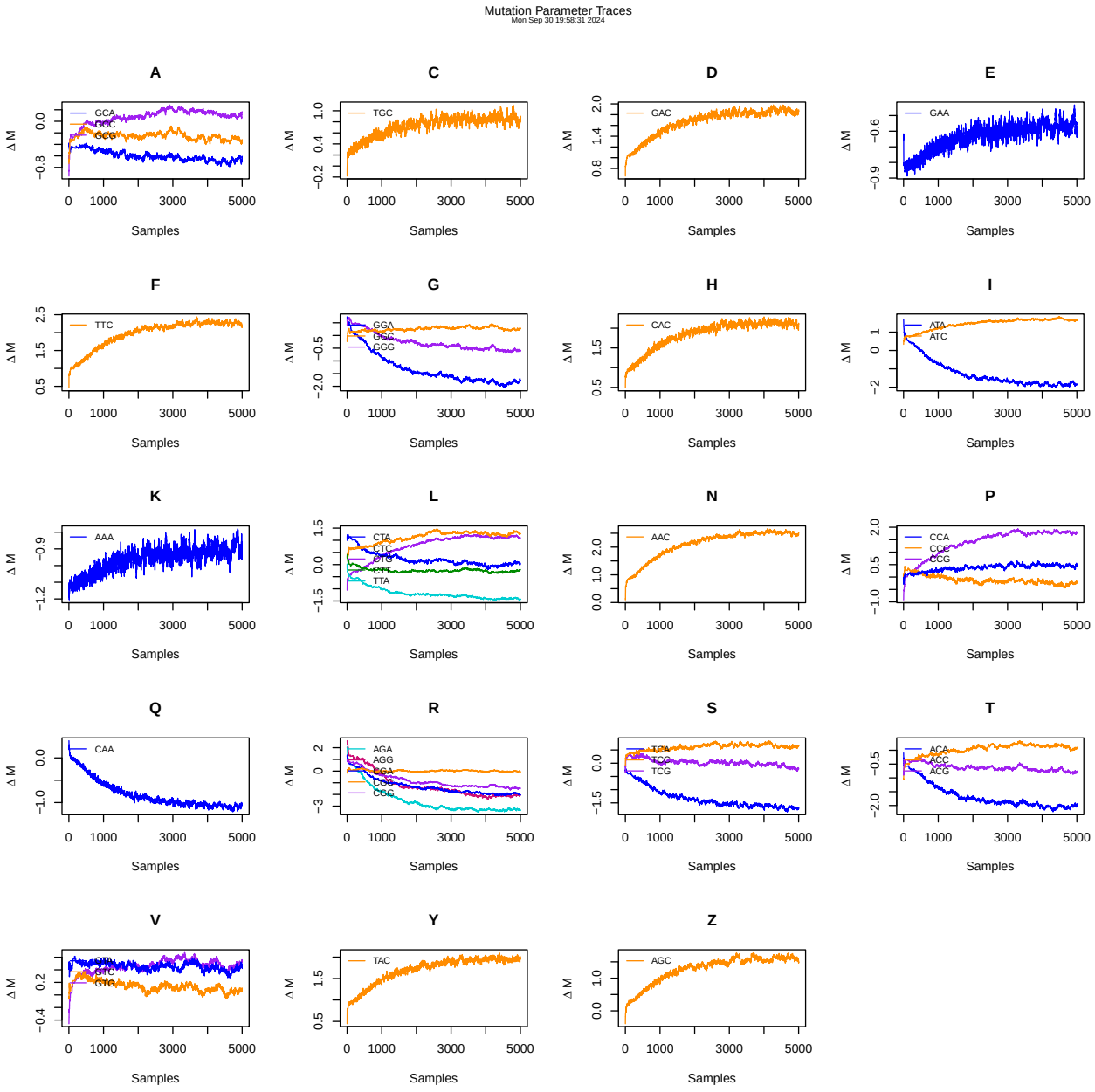
```
mcmc <- loadMCMCObject(file = "mcmc_out_Ecoli_split_sharedSelection_sphi1_09302024.Rda")

files <- "parameter_out_Ecoli_split_sharedSelection_sphi1_09302024.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.

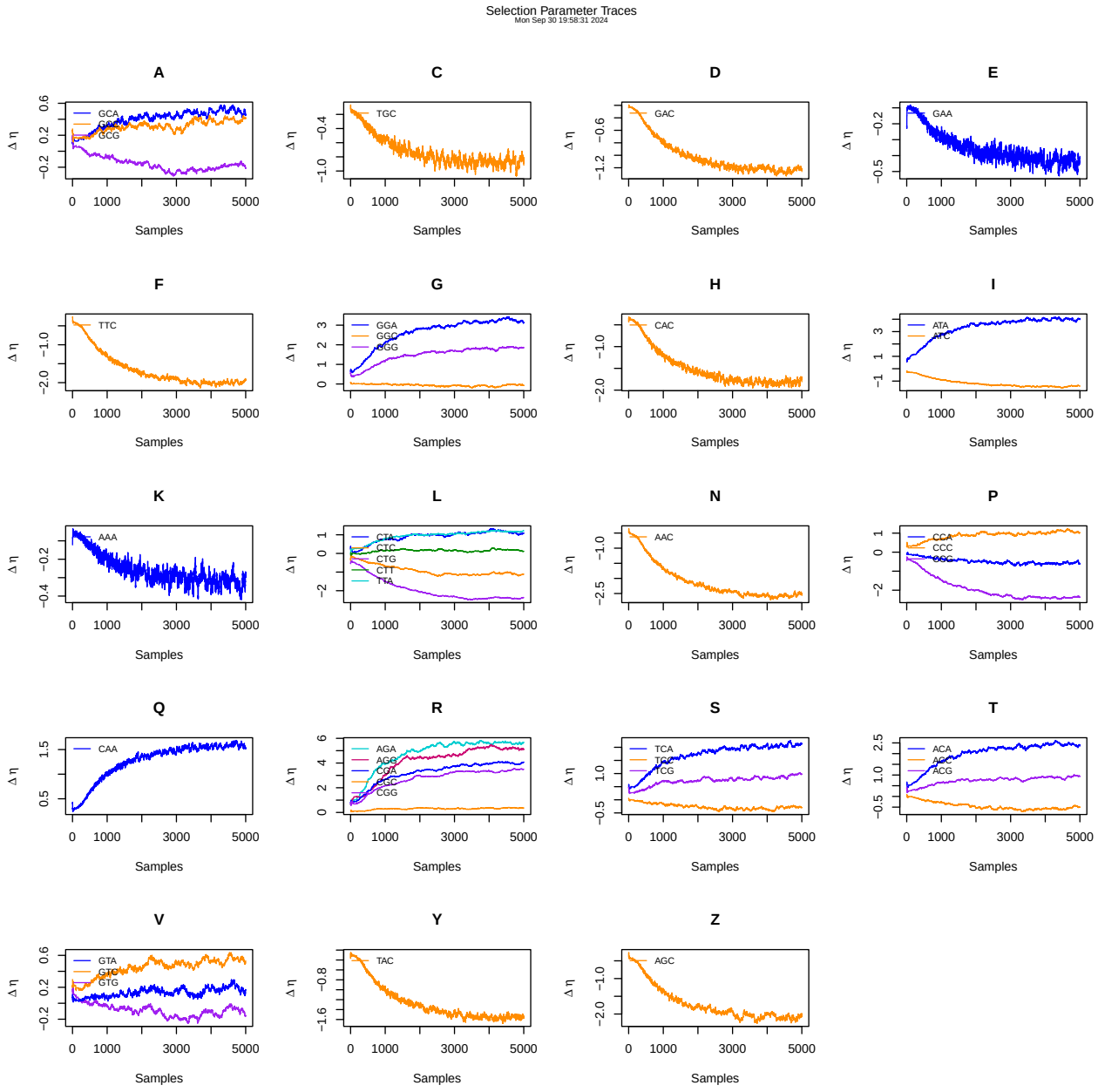
```
trace <- getTrace(parameter)  #store the trace (i.e. record of all iterations)

#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)
```



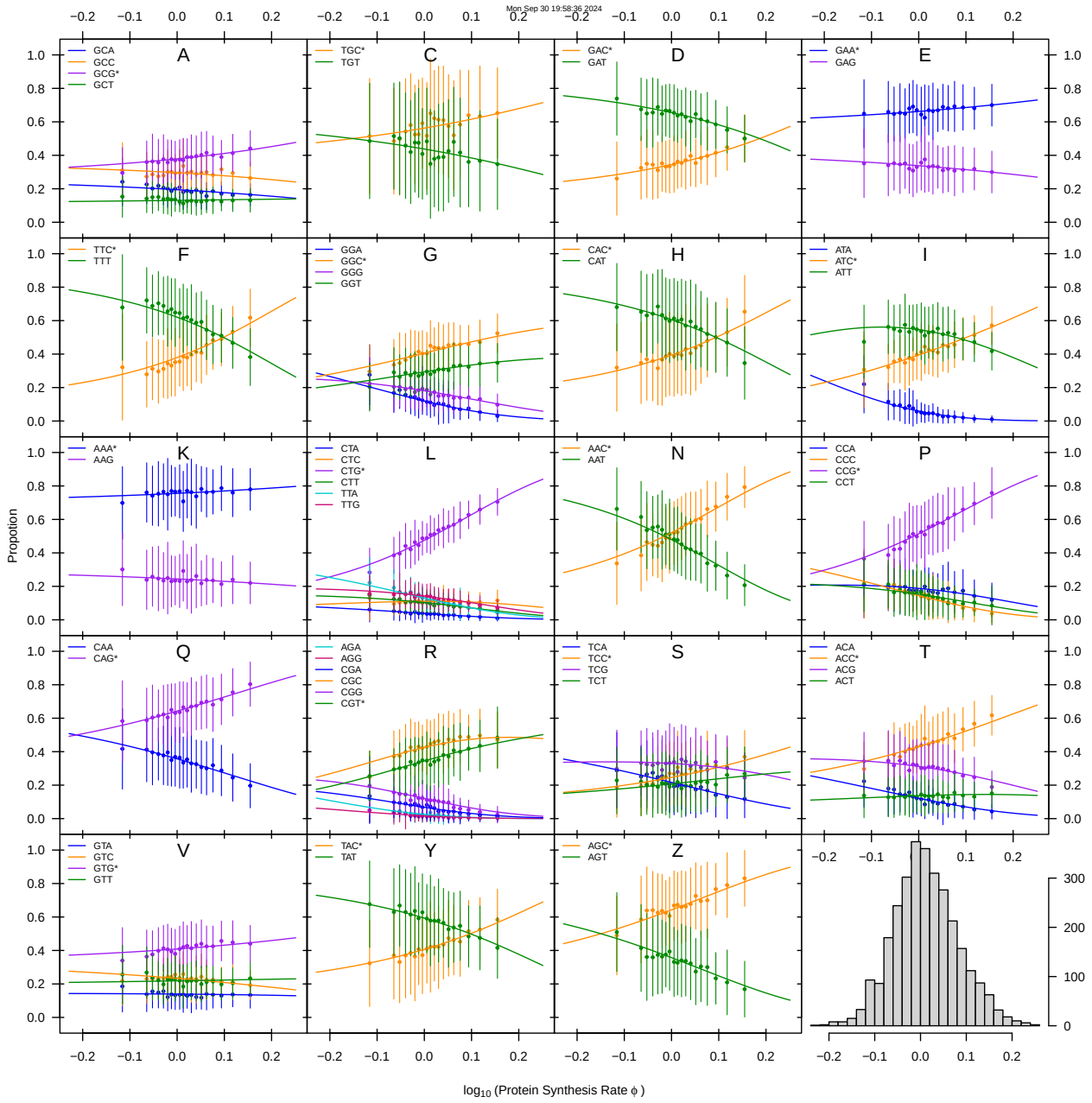

```
#We can also visualize the selection parameter
#Since we fixed DEta, it should be a single straight line.
plot.diag.1s <- plot(x = trace, what = "Selection", mixture = 1)
```



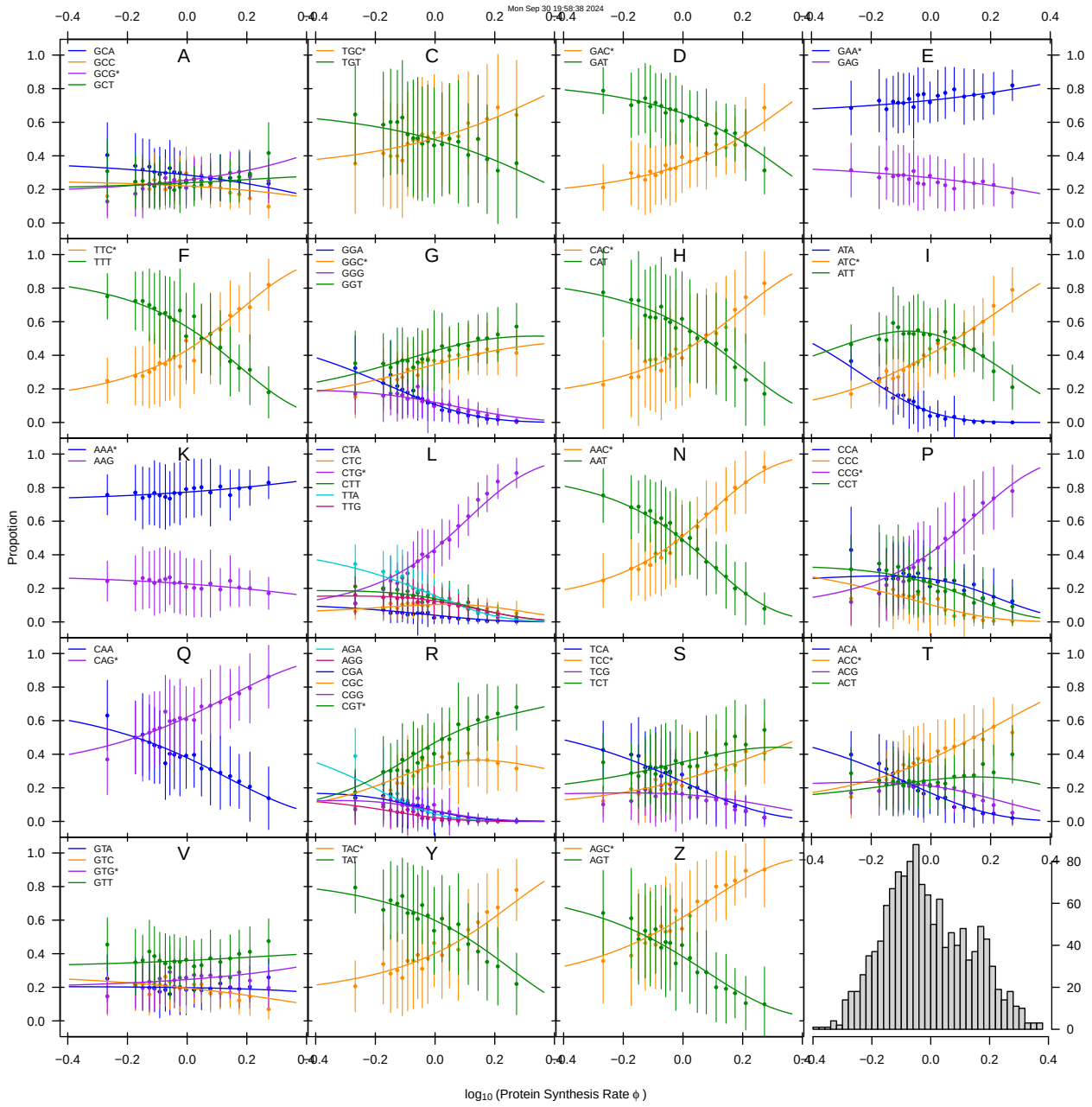

```
plot.diag.2s <- plot(x = trace, what = "Selection", mixture = 2)
```

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)
```

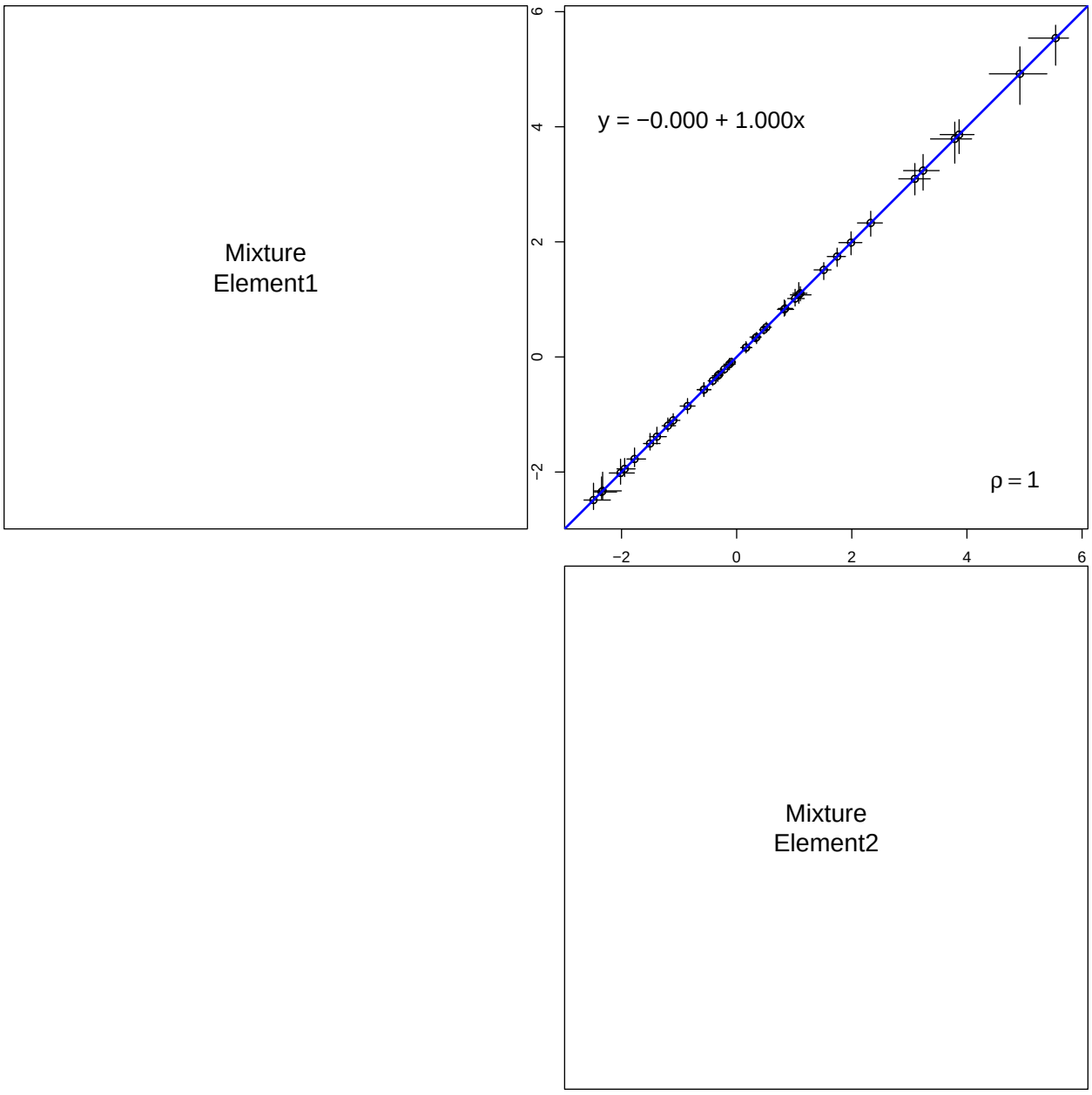



```
plot(x = model, genome = genomes, samples = 3000, mixture = 2)
```

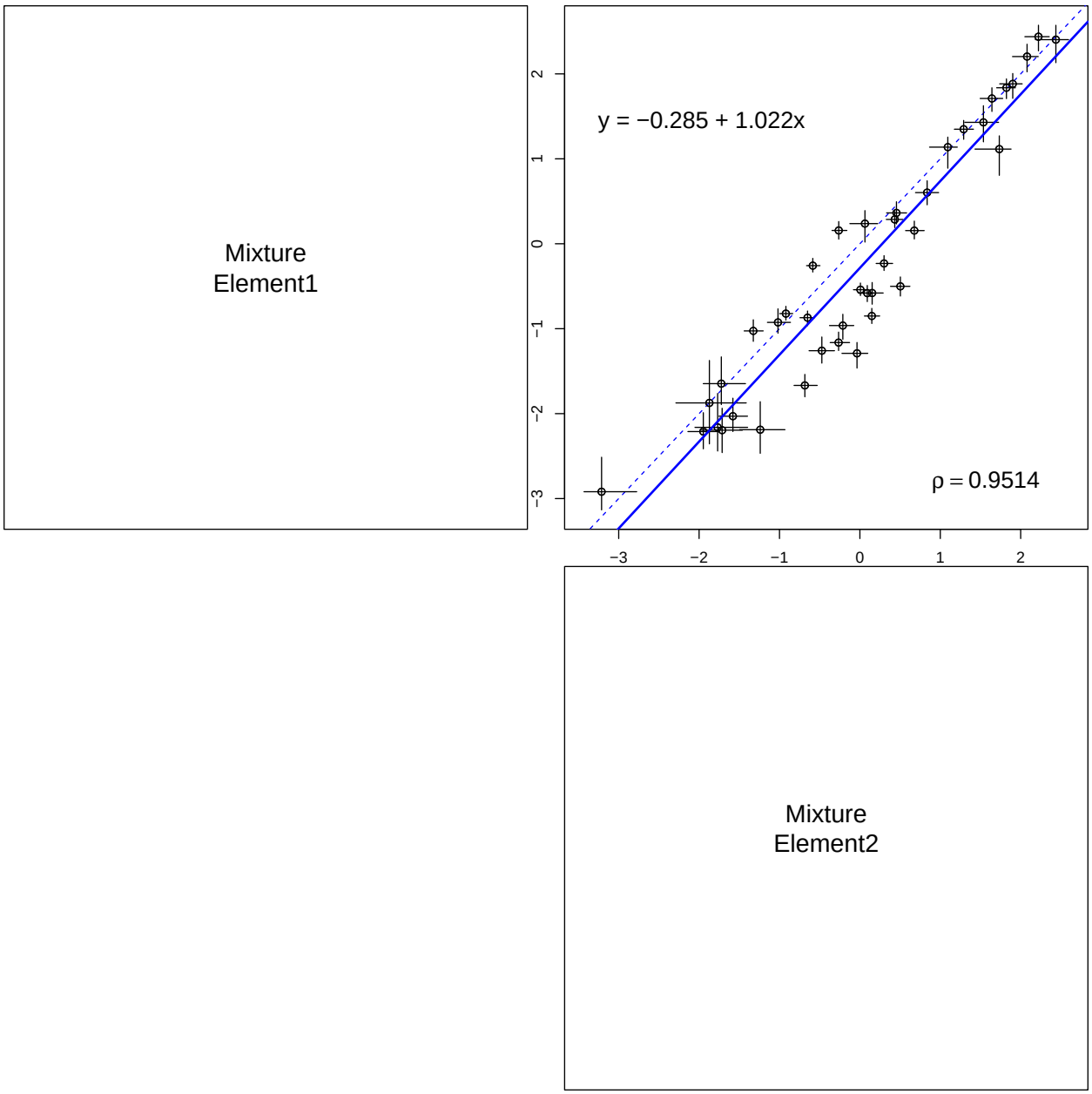



```
#This will compare different 'mixtures' i.e. gene sets
plot(parameter, what = "Selection", samples = 3000)
```

```
## 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
```



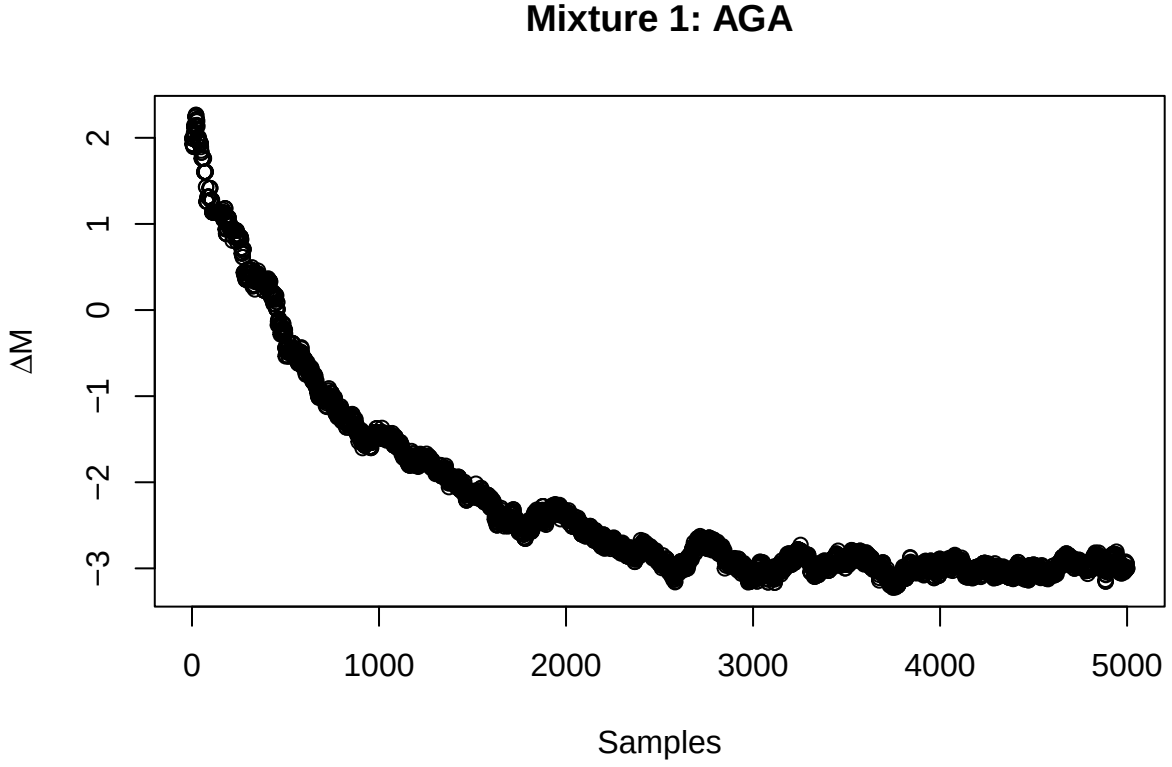

```
plot(parameter, what = "Mutation", samples = 3000)
```



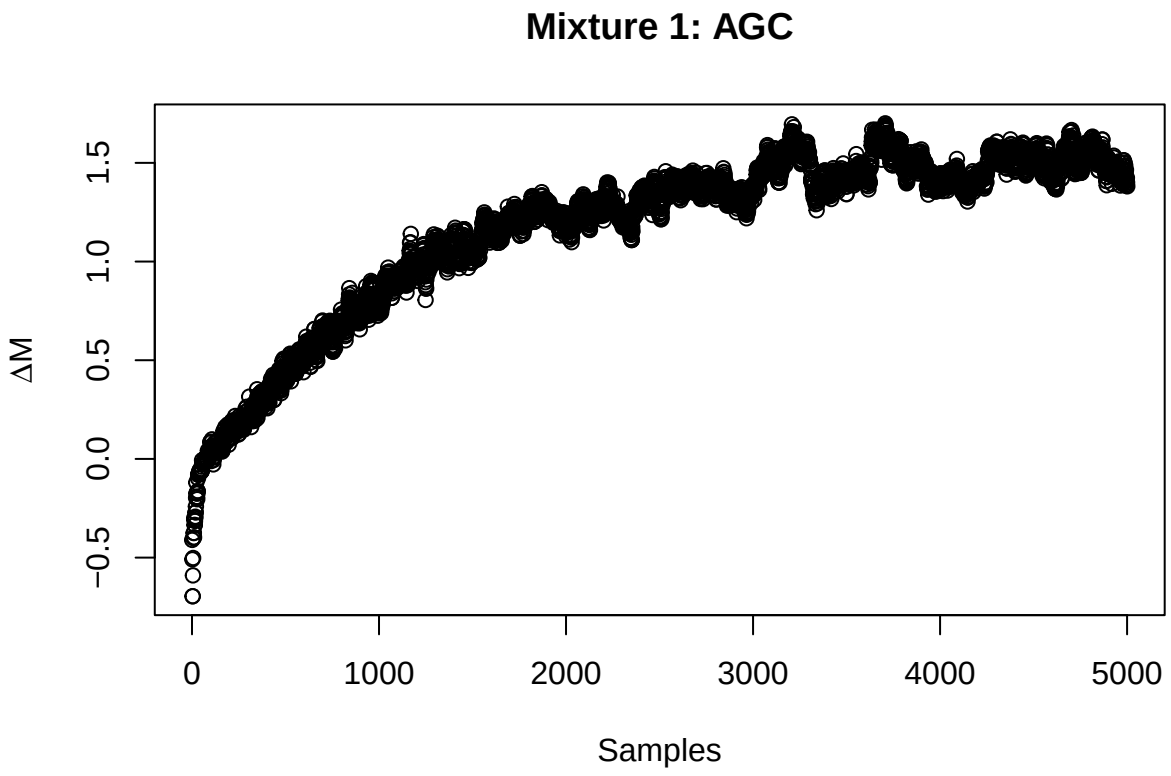
Now, we want to extract the mutation trace for each individual codon.

```
#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
```

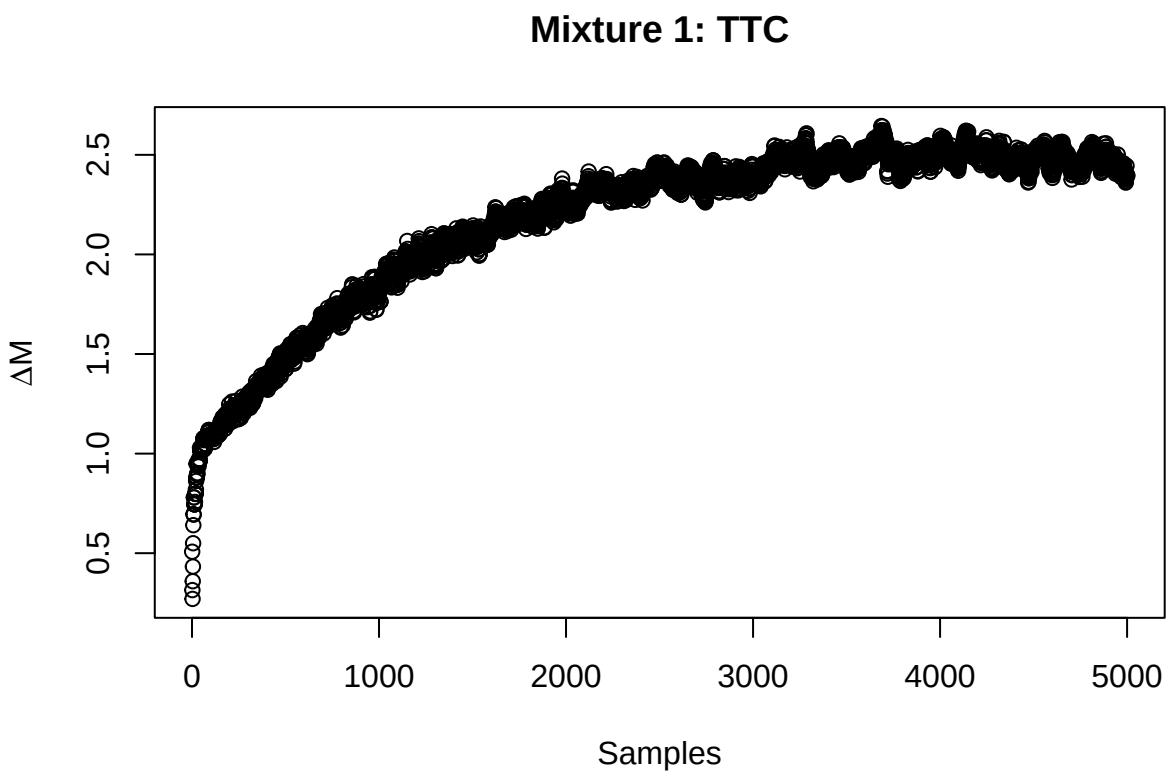
```
  data.frame(aa = names.aa) %>%        #make a dataframe
  select(-.) %>%                       #remove duplicated column
  mutate(Codon = lapply(aa, AAToCodon)) %>% #make new column with associated codons for each amino acid
  mutate(Codon = map(Codon, ~ .x[-length(.x)])) %>% #remove the reference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



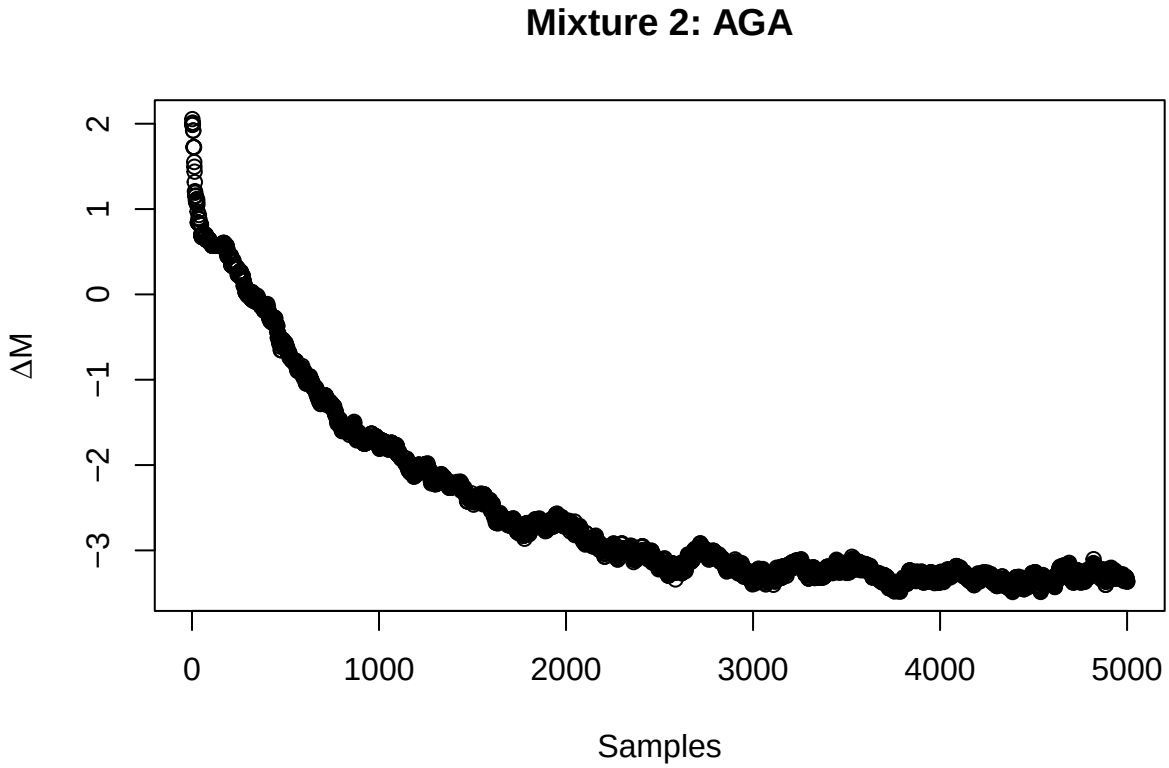

```
plot(mutationTrace$mix1$AGC, main = "Mixture 1: AGC", ylab = expression(Delta * M), xlab = "Samples")
```



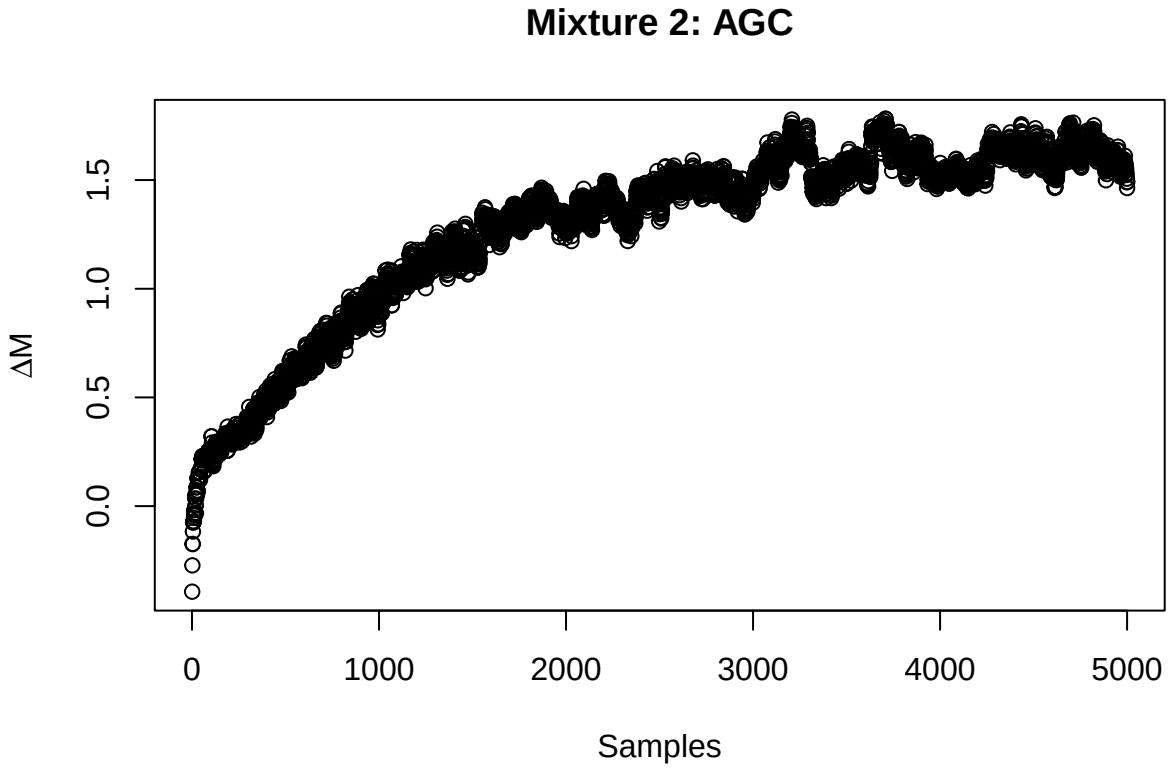

```
plot(mutationTrace$mix1$TTC, main = "Mixture 1: TTC", ylab = expression(Delta * M), xlab = "Samples")
```



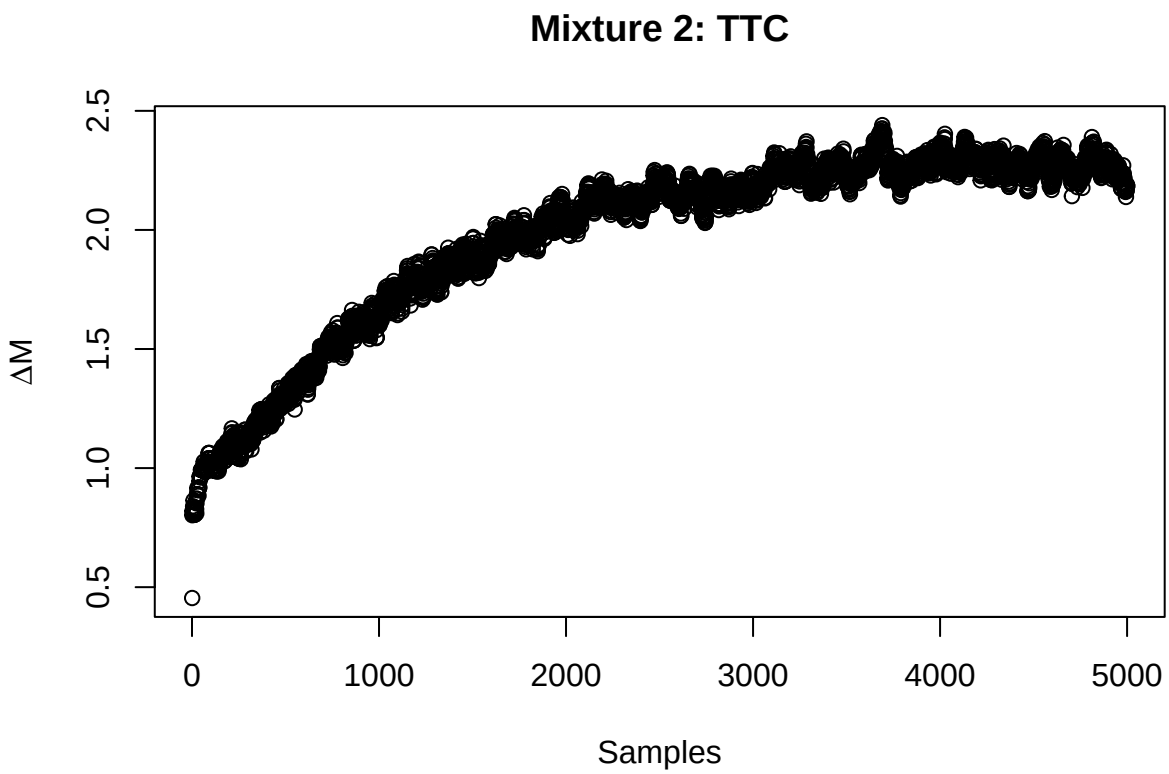
```
plot(mutationTrace$mix2$AGA, main = "Mixture 2: AGA", ylab = expression(Delta * M), xlab = "Samples")
```




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

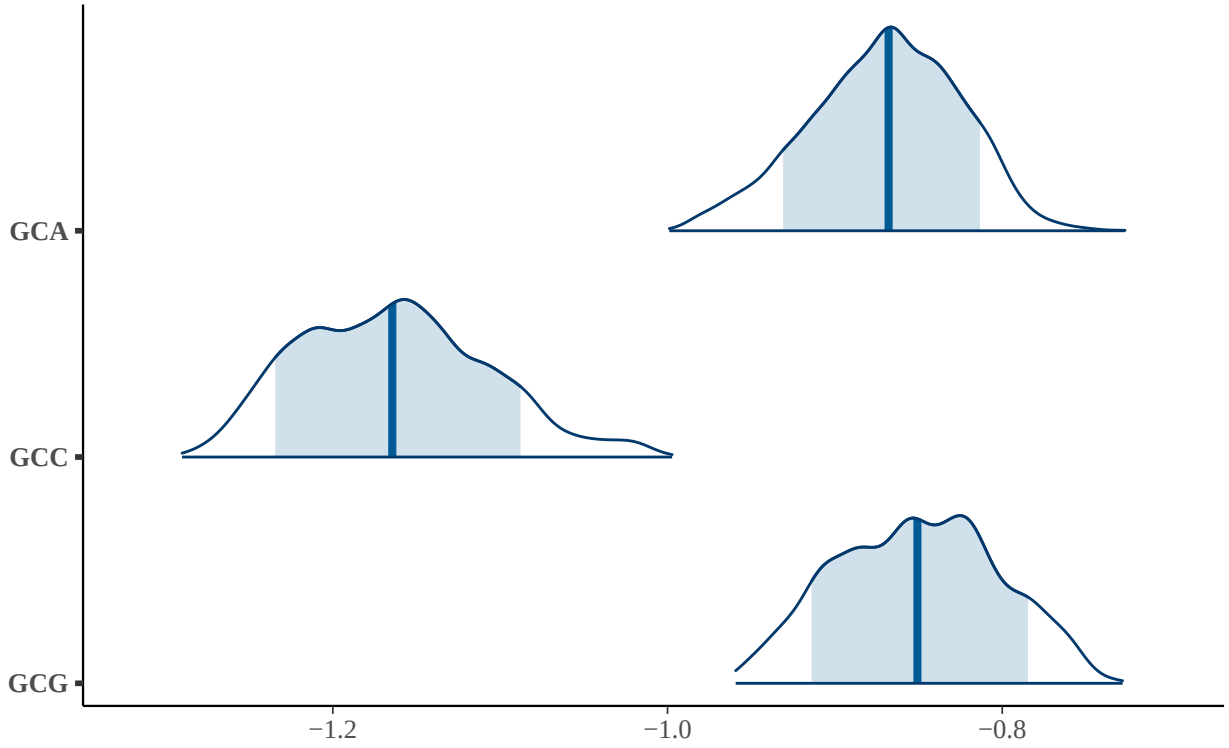


```r
plot(mutationTrace$mix2$TTC, main = "Mixture 2: TTC", ylab = expression(Delta * M), xlab = "Samples")
```

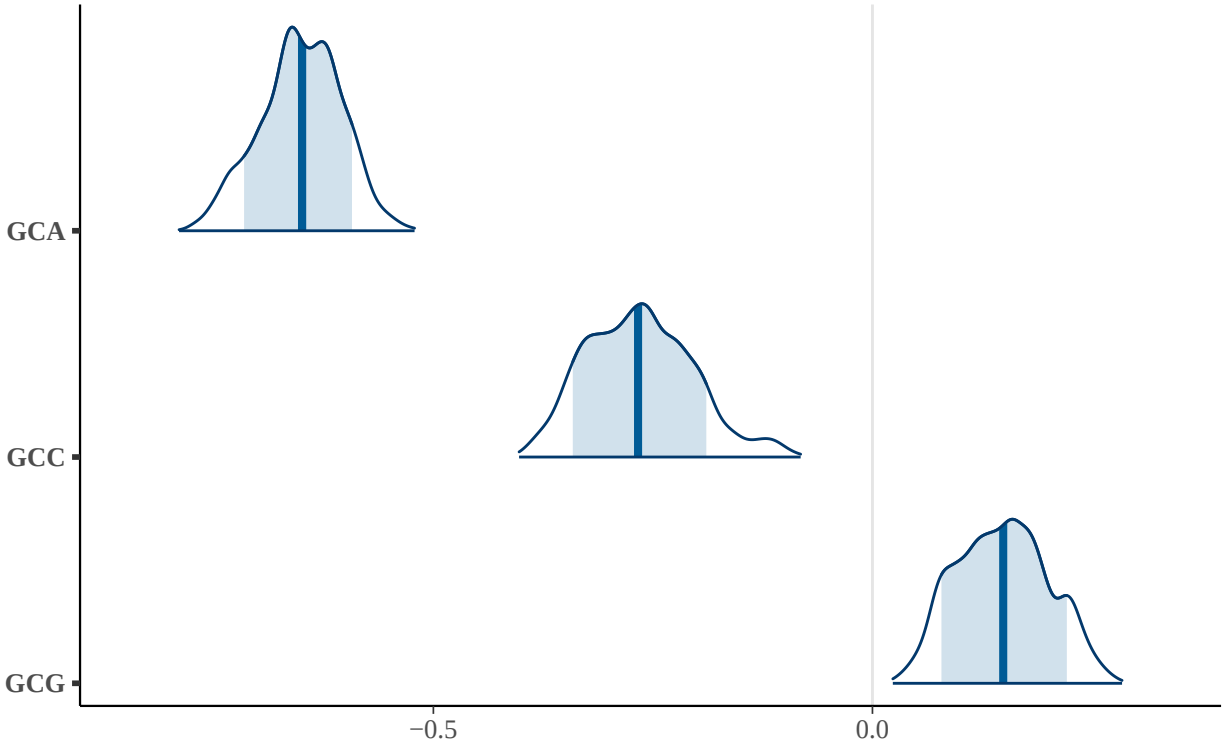


Now that we have the posterior distributions of our estimated parameter, let's try to plot it using Bayesplot.

```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")# +
```



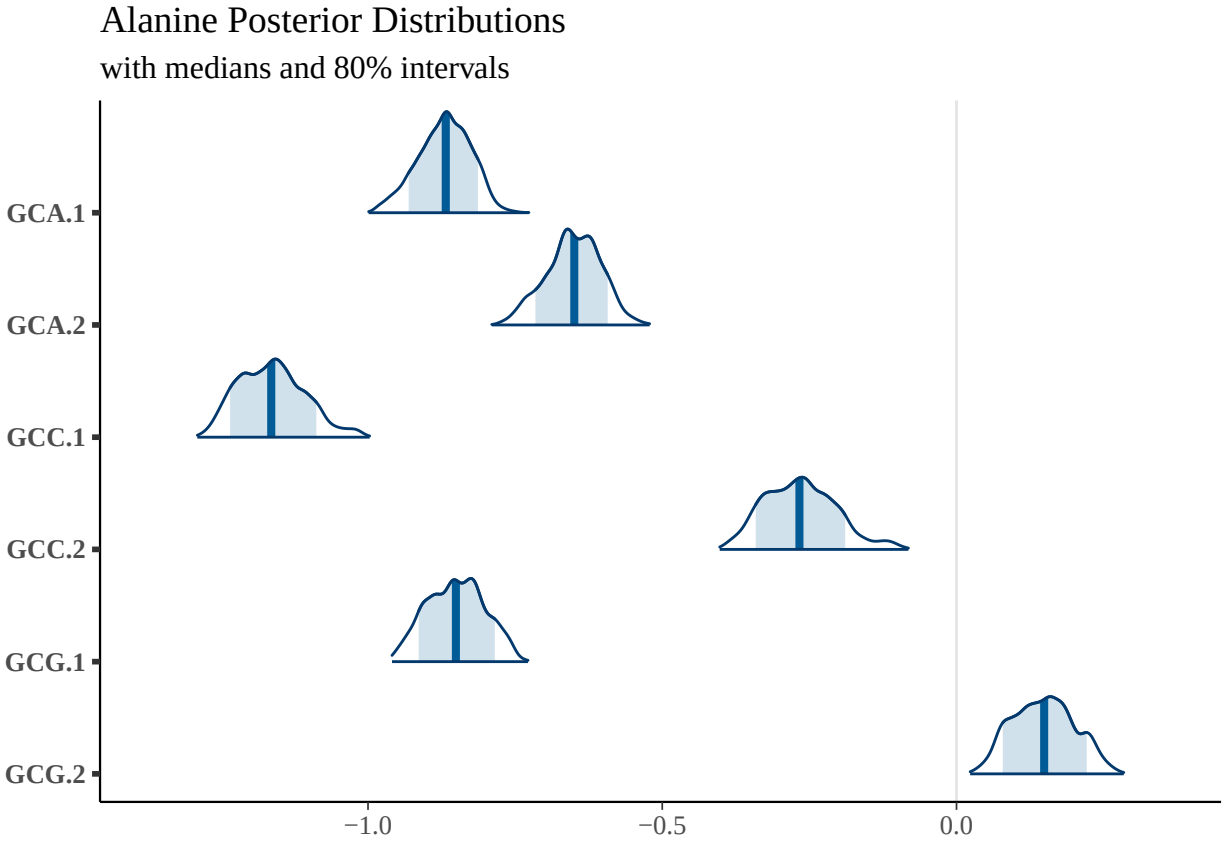

```
  #coord_cartesian(xlim = c(-1.5, 0.5))                    #adjust x-axis

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

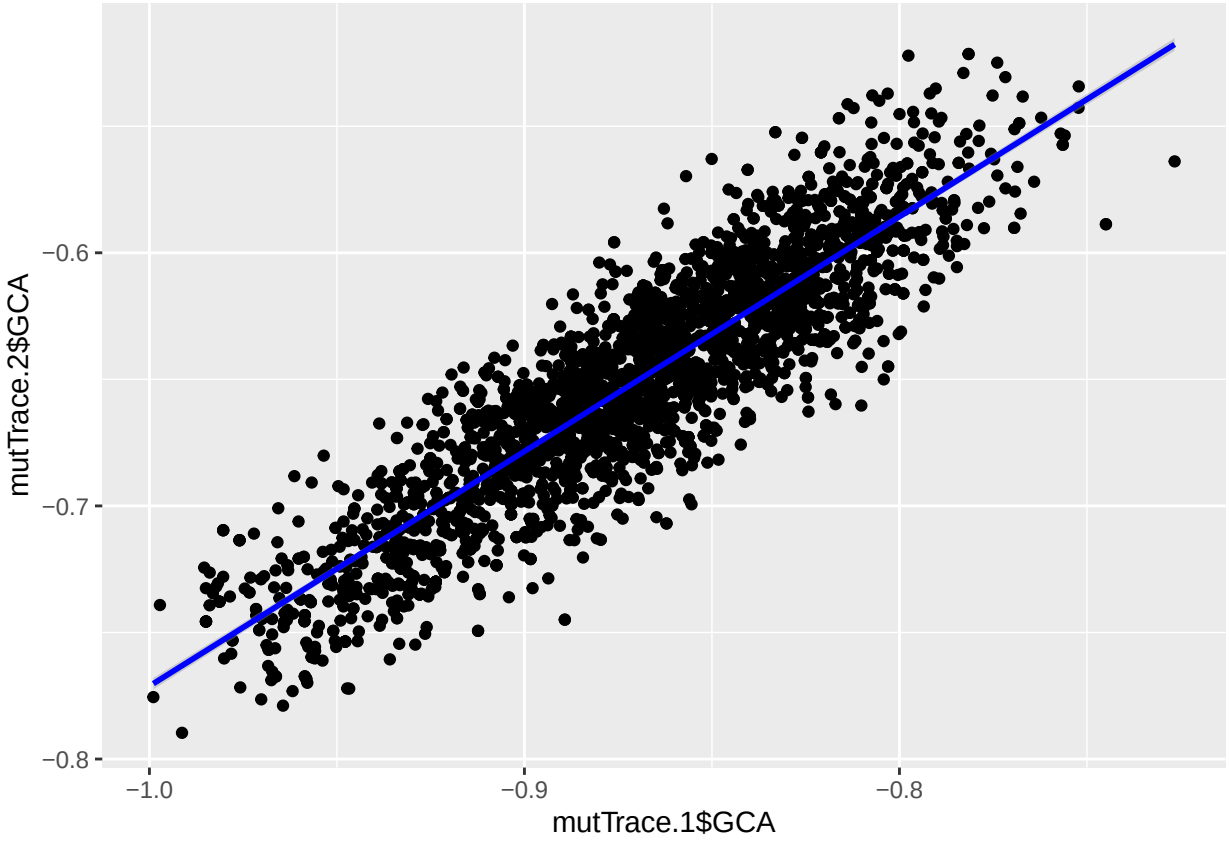
```
#Example: show all codons of alanine across the mixtures
alanine <- data.frame(
  GCA.1 = mutTrace.1$GCA,
  GCA.2 = mutTrace.2$GCA,
  GCC.1 = mutTrace.1$GCC,
  GCC.2 = mutTrace.2$GCC,
  GCG.1 = mutTrace.1$GCG,
  GCG.2 = mutTrace.2$GCG
)

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'
```

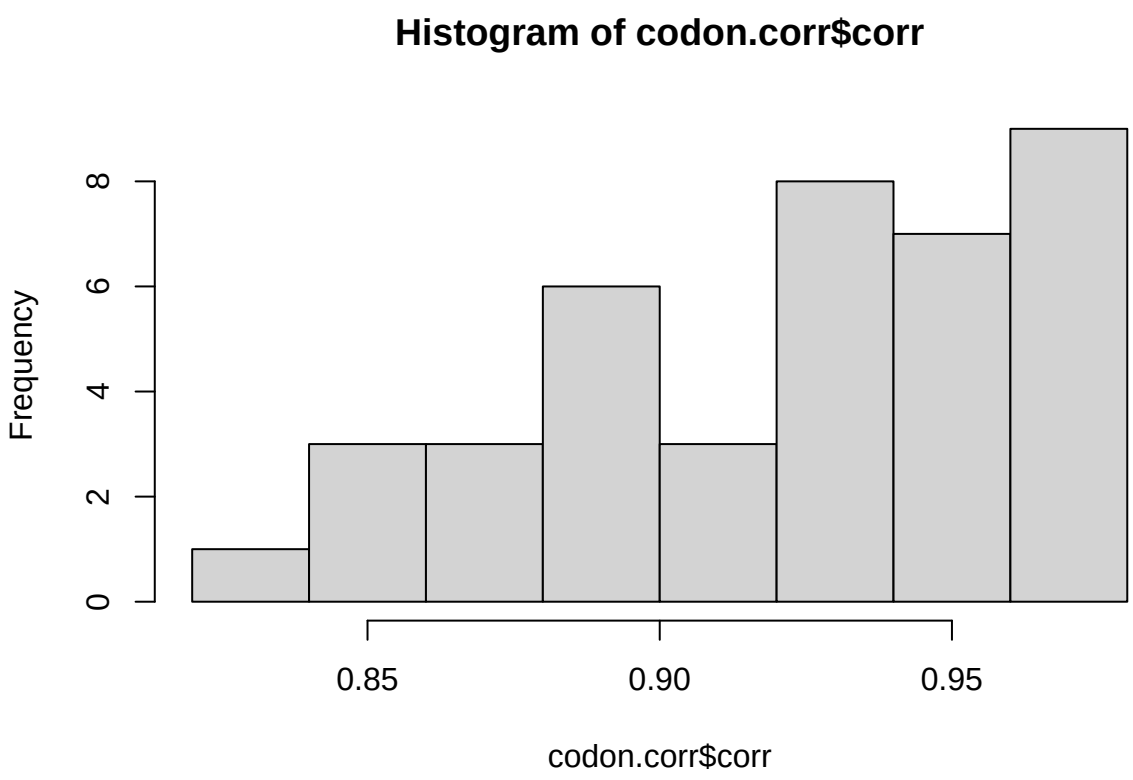



```
regression <- lm(mutTrace.2$GCA ~ mutTrace.1$GCA)
summary(regression)
```

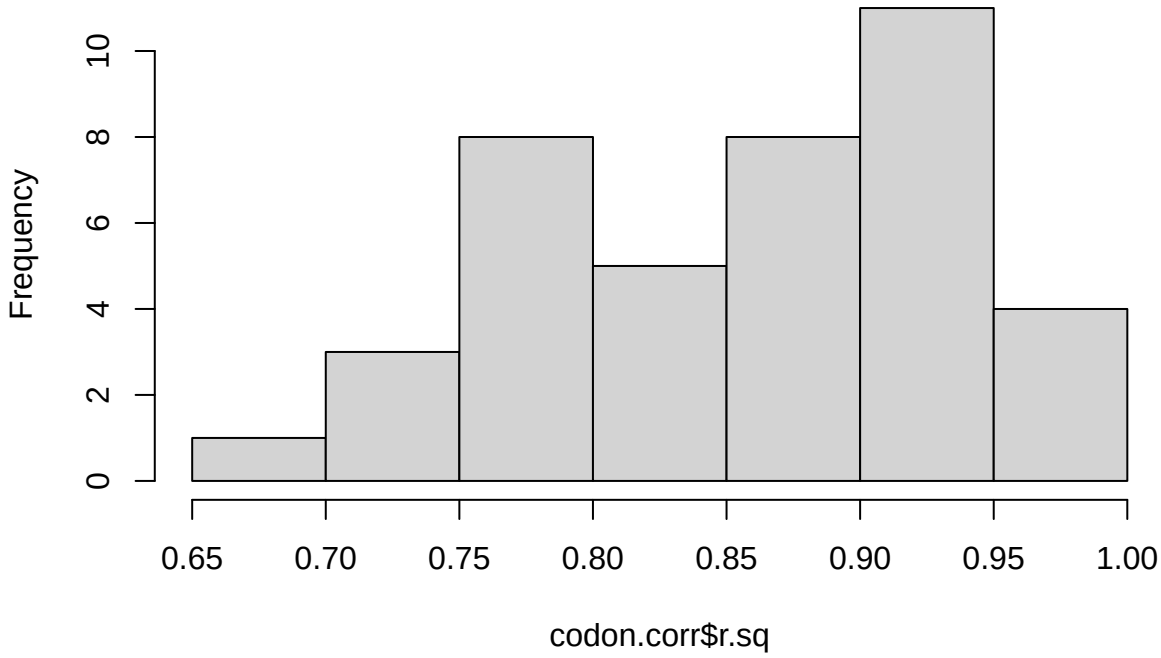
```
##
## Call:
## lm(formula = mutTrace.2$GCA ~ mutTrace.1$GCA)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.076464 -0.014076  0.000084  0.014629  0.069225
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.156138   0.007479   20.88   <2e-16 ***
## mutTrace.1$GCA 0.927326   0.008582  108.05   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02087 on 2999 degrees of freedom
## Multiple R-squared:  0.7956, Adjusted R-squared:  0.7956
## F-statistic: 1.167e+04 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 0.8919759 0.7956209
## GCC   GCC 0.9273490 0.8599761
## GCG   GCG 0.9140621 0.8355095
## TGC   TGC 0.8464468 0.7164722
## GAC   GAC 0.9383736 0.8805450
## GAA   GAA 0.8873132 0.7873247
## TTC   TTC 0.9560270 0.9139877
## GGA   GGA 0.9799531 0.9603080
## GGC   GGC 0.9322903 0.8691653
## GGG   GGG 0.9590507 0.9197782
## CAC   CAC 0.9300659 0.8650227
## ATA   ATA 0.9715918 0.9439906
## ATC   ATC 0.9609263 0.9233794
## AAA   AAA 0.8467031 0.7169061
## CTA   CTA 0.8922277 0.7960702
## CTC   CTC 0.8929408 0.7973433
## CTG   CTG 0.9687175 0.9384136
## CTT   CTT 0.8263648 0.6828788
## TTA   TTA 0.9360570 0.8762027
## AAC   AAC 0.9776285 0.9557574
## CCA   CCA 0.8759475 0.7672840
## CCC   CCC 0.8420558 0.7090580
## CCG   CCG 0.9607282 0.9229988
## CAA   CAA 0.9563322 0.9145713
## AGA   AGA 0.9235270 0.8529021
## AGG   AGG 0.9771317 0.9547864
```

```
## CGA   CGA 0.9770833 0.9546918
## CGC   CGC 0.8965443 0.8037917
## CGG   CGG 0.9690656 0.9390880
## TCA   TCA 0.9418867 0.8871505
## TCC   TCC 0.8711688 0.7589350
## TCG   TCG 0.9207420 0.8477659
## ACA   ACA 0.9531292 0.9084553
## ACC   ACC 0.9150146 0.8372517
## ACG   ACG 0.9189653 0.8444972
## GTA   GTA 0.8921730 0.7959727
## GTC   GTC 0.8799746 0.7743554
## GTG   GTG 0.9490954 0.9007821
## TAC   TAC 0.9379971 0.8798386
## AGC   AGC 0.9596288 0.9208874
```

```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 0.9273257 0.008582408 108.04959       0
## GCC   GCC 1.0031507 0.007391556 135.71576       0
## GCG   GCG 0.9859227 0.007988207 123.42228       0
## TGC   TGC 0.8491185 0.009753898  87.05427       0
## GAC   GAC 0.9687924 0.006515816 148.68321       0
## GAA   GAA 0.9350576 0.008874251 105.36749       0
## TTC   TTC 0.9482809 0.005312010 178.51639       0
## GGA   GGA 0.9674903 0.003591736 269.36562       0
## GGC   GGC 1.0833981 0.007675562 141.14902       0
## GGG   GGG 0.9824507 0.005298186 185.43154       0
## CAC   CAC 0.9350771 0.006744906 138.63457       0
## ATA   ATA 0.9437537 0.004197753 224.82352       0
## ATC   ATC 0.9548822 0.005022783 190.11020       0
## AAA   AAA 0.8789252 0.010085508  87.14733       0
## CTA   CTA 0.8781761 0.008116299 108.19909       0
## CTC   CTC 0.9492420 0.008738695 108.62515       0
## CTG   CTG 0.9491926 0.004440289 213.76820       0
## CTT   CTT 0.7450607 0.009271378  80.36138       0
## TTA   TTA 0.9227036 0.006333258 145.69177       0
## AAC   AAC 0.9419981 0.003700908 254.53163       0
## CCA   CCA 0.8442831 0.008490531  99.43819       0
```

```
## CCC   CCC 0.8525259 0.009971986  85.49209       0
## CCG   CCG 0.9310716 0.004910697 189.60071       0
## CAA   CAA 0.9584151 0.005348828 179.18227       0
## AGA   AGA 0.9500393 0.007204553 131.86652       0
## AGG   AGG 0.8653678 0.003438699 251.65559       0
## CGA   CGA 0.9312128 0.003704396 251.38045       0
## CGC   CGC 1.0695294 0.009649202 110.84122       0
## CGG   CGG 0.8888465 0.004133678 215.02558       0
## TCA   TCA 0.9164376 0.005968509 153.54550       0
## TCC   TCC 0.9233175 0.009502275  97.16804       0
## TCG   TCG 0.9595464 0.007424989 129.23203       0
## ACA   ACA 0.9123012 0.005288283 172.51368       0
## ACC   ACC 0.9591693 0.007722131 124.21044       0
## ACG   ACG 0.9534956 0.007471374 127.61985       0
## GTA   GTA 0.9612917 0.008887140 108.16661       0
## GTC   GTC 0.9650664 0.009512865 101.44856       0
## GTG   GTG 1.0309289 0.006247777 165.00730       0
## TAC   TAC 0.9280018 0.006262411 148.18603       0
## AGC   AGC 0.9339299 0.004998567 186.83952       0
```