

# 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
## vforcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyverse 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(flag==1){                           #if row starts with >
        sequence<-str_c(alist,collapse = "") #flag check to concatenate sequence lines
        aasize<-nchar(sequence)              #collapse
                                                #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.chr)          #find and store species name (user supplies species name)
      b<-str_extract(i,pattern.gn)           #find and store gene name
      c<-str_extract(i,pattern.type)         #find and store pattern type (>tr or >sp, should be only
                                              #make above identifiers into easy to bind object
      abci<-c(a,b,c,i)                     #rbind to fill rows of new df
      y<-rbind(y,abci)
    }
    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")
loadROCPParameterObject <- 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:max]
}
#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$paramBase$synthesisOffsetAcceptanceRateTrace
curObservedSynthesisNoiseTrace <- tempEnv$paramBase$observedSynthesisNoiseTrace

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

curStdDevSynthesisRateTraces <- tempEnv$paramBase$stdDevSynthesisRateTraces
curStdDevSynthesisRateAcceptanceRateTrace <- tempEnv$paramBase$stdDevSynthesisRateAcceptanceRateTrace
curSynthesisRateTrace <- tempEnv$paramBase$synthesisRateTrace
curSynthesisRateAcceptanceRateTrace <- tempEnv$paramBase$synthesisRateAcceptanceRateTrace
curMixtureAssignmentTrace <- tempEnv$paramBase$mixtureAssignmentTrace
curMixtureProbabilitiesTrace <- tempEnv$paramBase$mixtureProbabilitiesTrace
curCodonSpecificAcceptanceRateTrace <- tempEnv$paramBase$codonSpecificAcceptanceRateTrace

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, curSynthesisRateAcceptanceRateTrace, max)

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

}

```

```

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](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+RESTART_5000_09302024.Rda")

files <- "parameter_out_Ecoli_split_sharedSelection_sphi1_RESTART_5000_09302024.Rda"

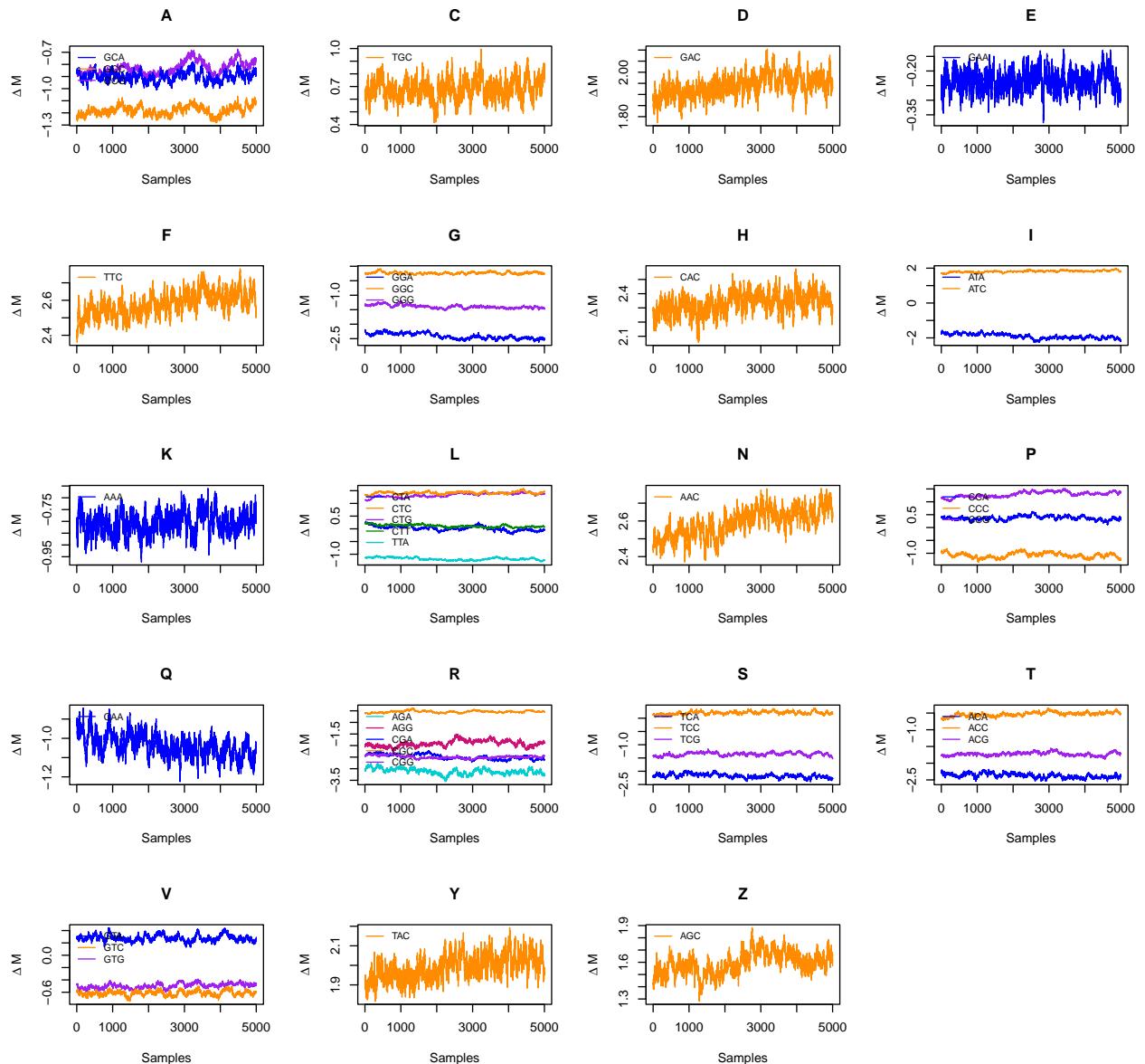
parameter <- loadROCPParameterObject(parameter, files)
```

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)

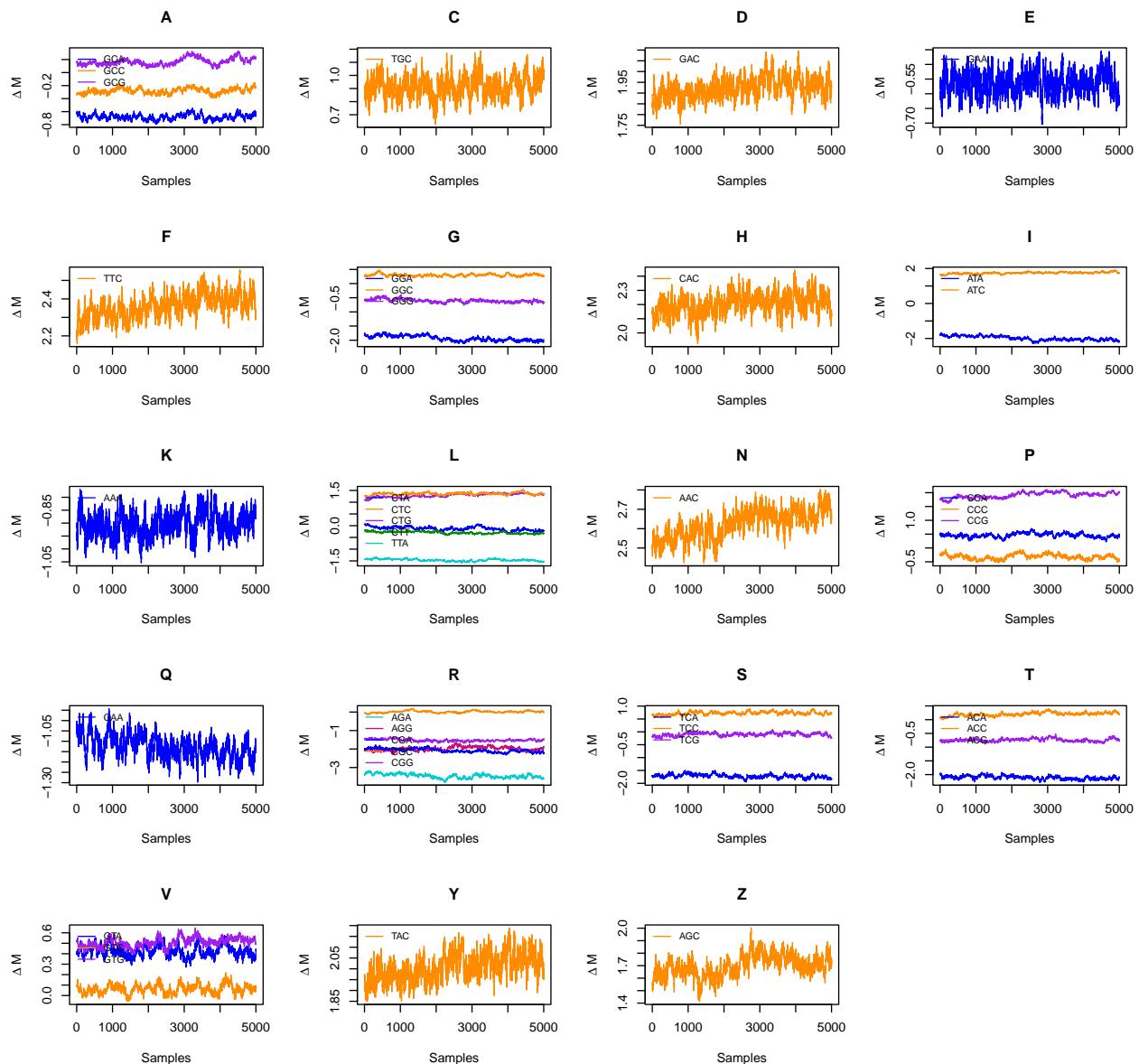
#Visualize the mutation parameters
#We are looking for the lines to level out.
plot.diag.1m <- plot(x = trace, what = "Mutation", mixture = 1)
```

Mutation Parameter Traces  
Tue Oct 1 10:22:48 2024



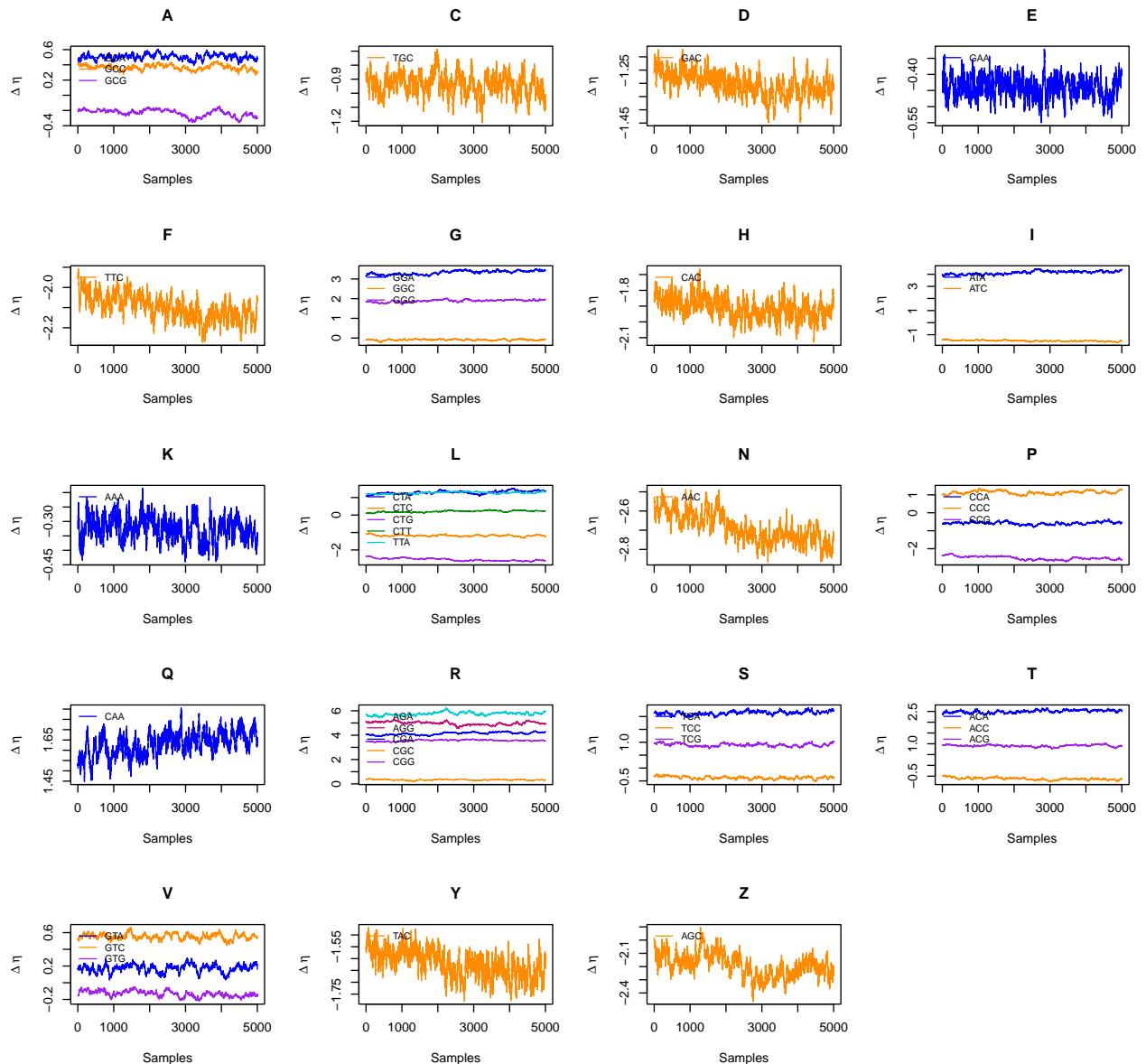
```
plot.diag.2m <- plot(x = trace, what = "Mutation", mixture = 2)
```

Mutation Parameter Traces  
Tue Oct 1 10:22:48 2024



```
#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)
```

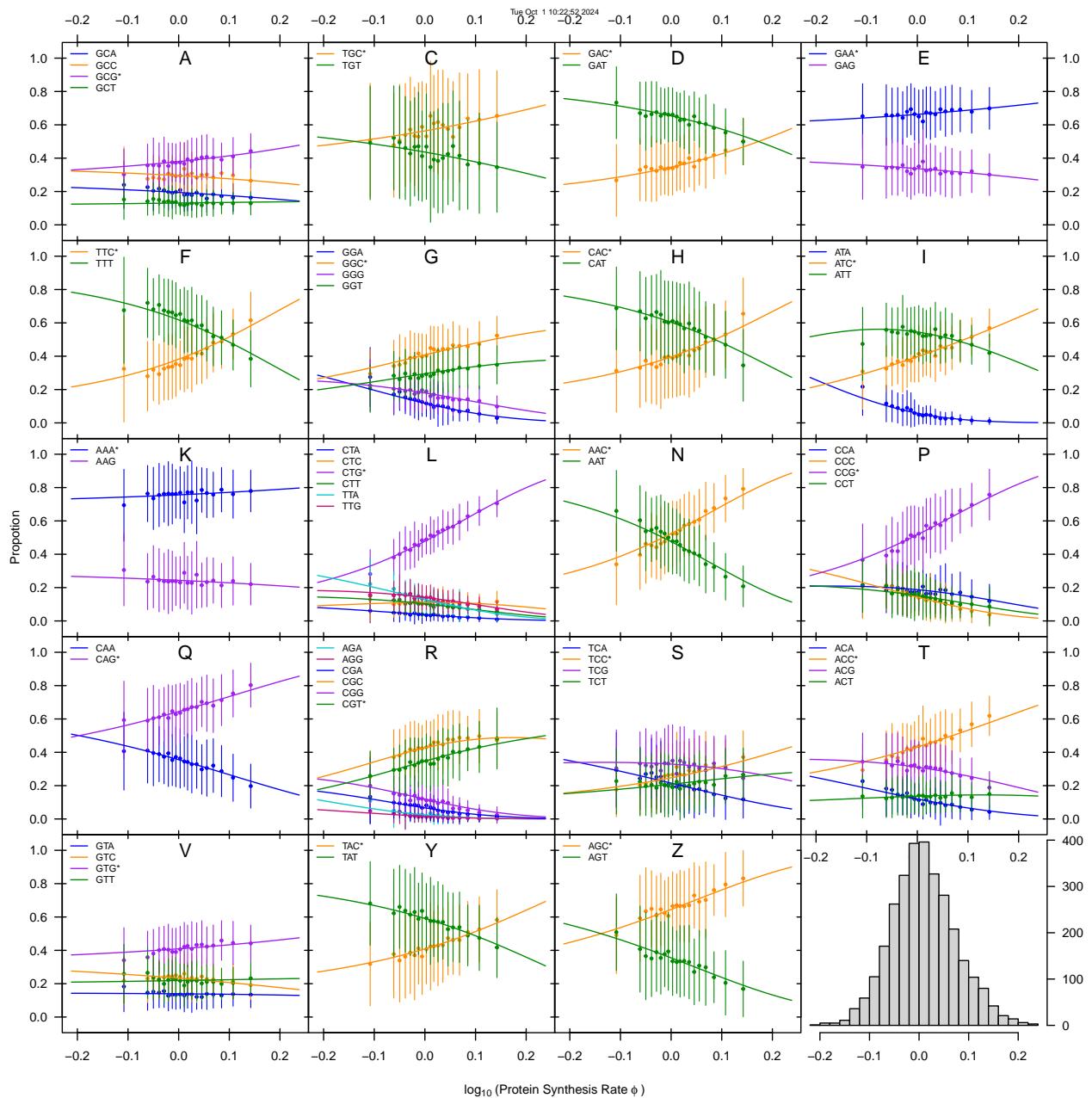
Selection Parameter Traces  
Tue Oct 1 16:22:48 2024



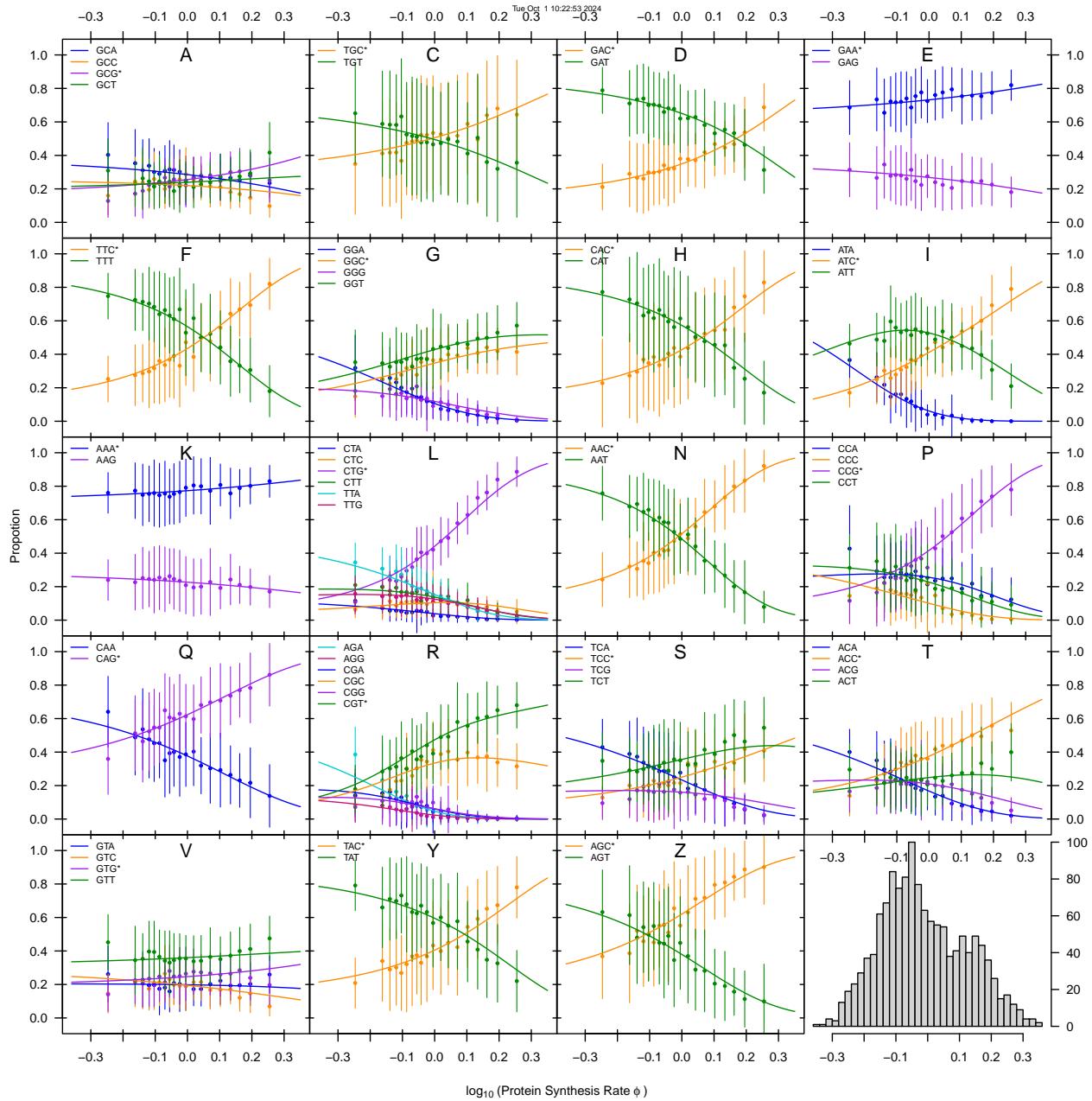
```
plot.diag.2s <- plot(x = trace, what = "Selection", mixture = 2)
```

The plots above should show each trace becoming more and more level. If not, increase the number of samples 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 = 5000, mixture = 1)
```

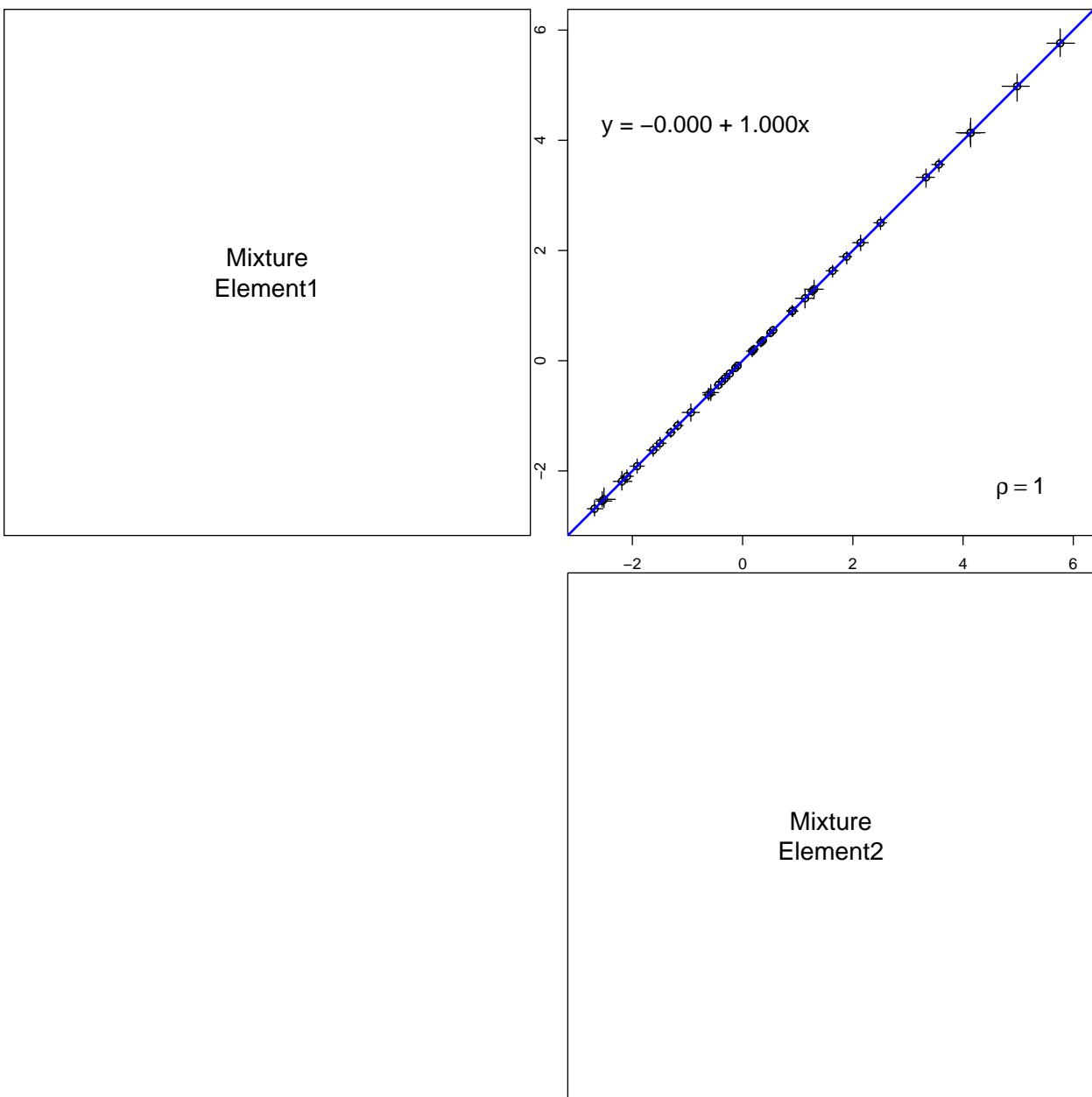


```
plot(x = model, genome = genomes, samples = 5000, mixture = 2)
```

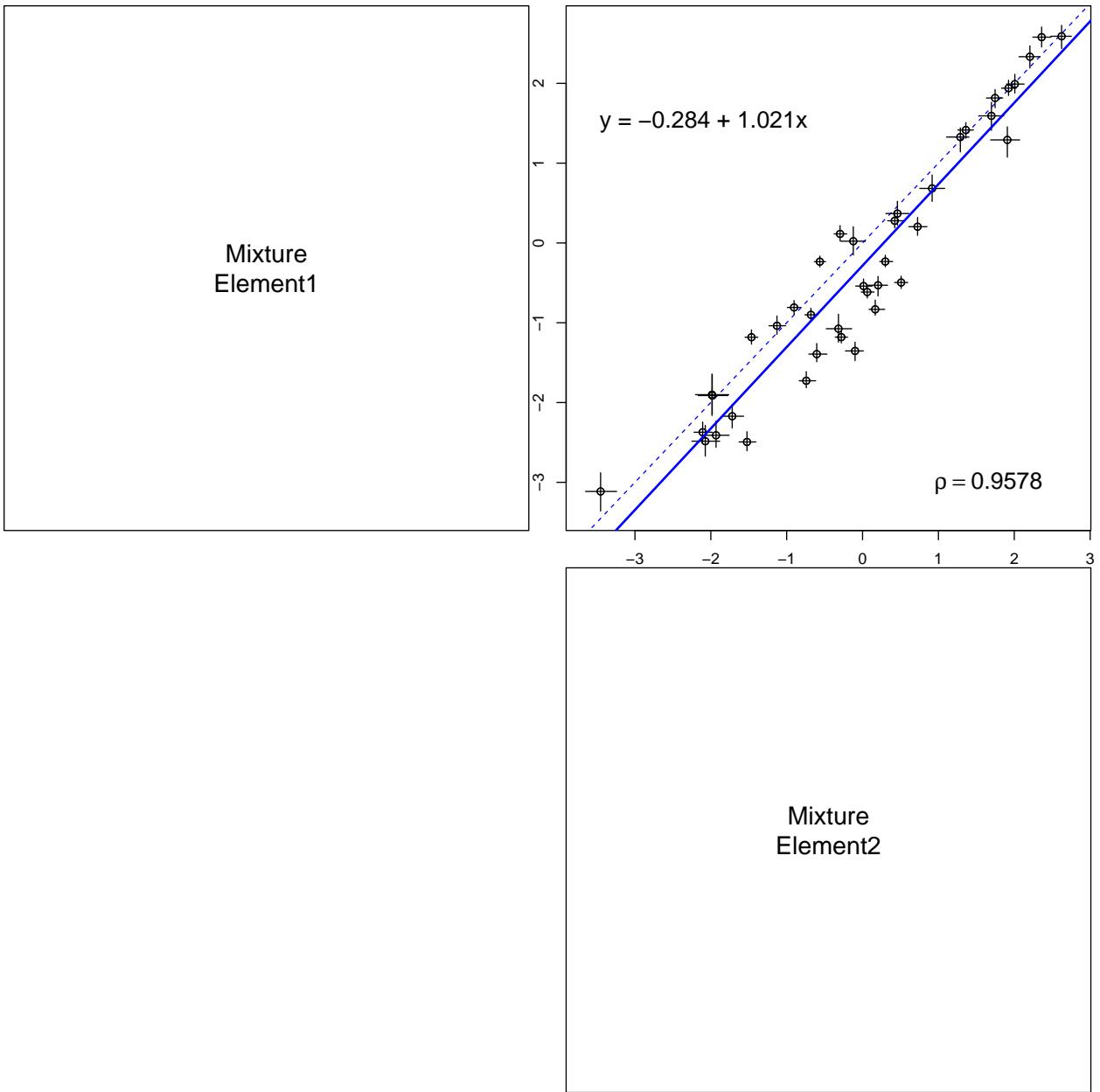


```
#This will compare different 'mixtures' i.e. gene sets
plot(parameter, what = "Selection", samples = 5000)
```

```
## 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 = 5000)
```



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
```

```

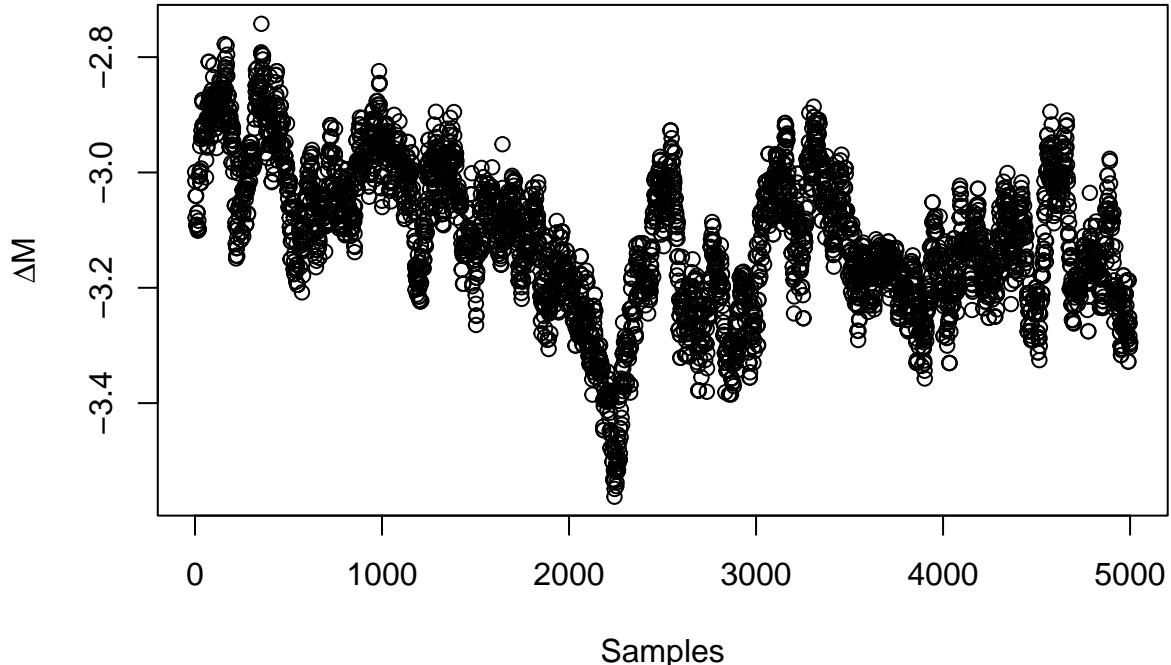
data.frame(aa = names_aa) %>%          #make a dataframe
select(-.) %>%                         #remove duplicated column
mutate(Codon = lapply(aa, AAToCodon)) %>% #make new column with associated codons for each amino acid
mutate(Codon = map(Codon, ~ .x[-length(.x)])) %>% #remove the reference codon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

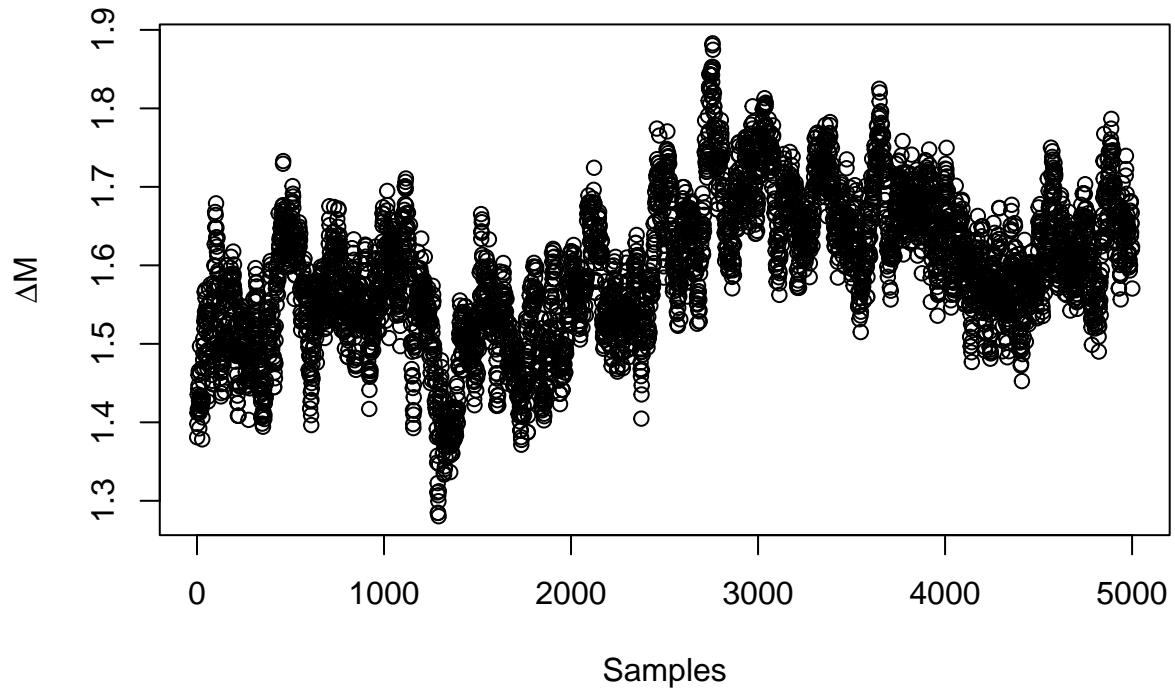
```

## Mixture 1: AGA



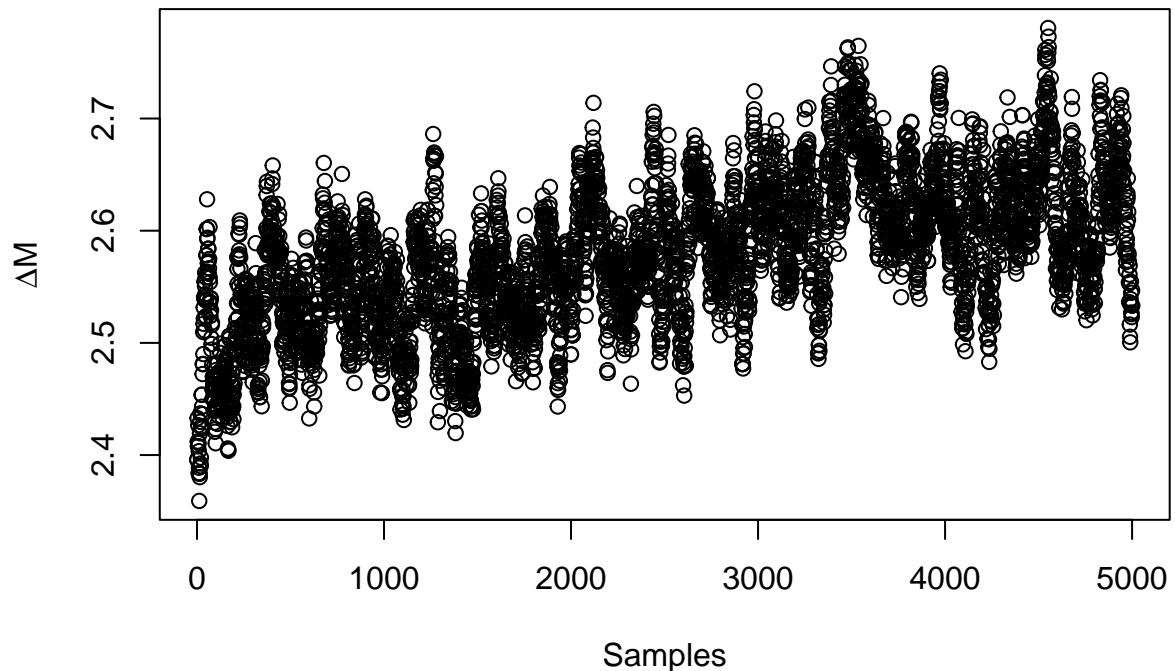
```
plot(mutationTrace$mix1$AGC, main = "Mixture 1: AGC", ylab = expression(Delta * M), xlab = "Samples")
```

### Mixture 1: AGC



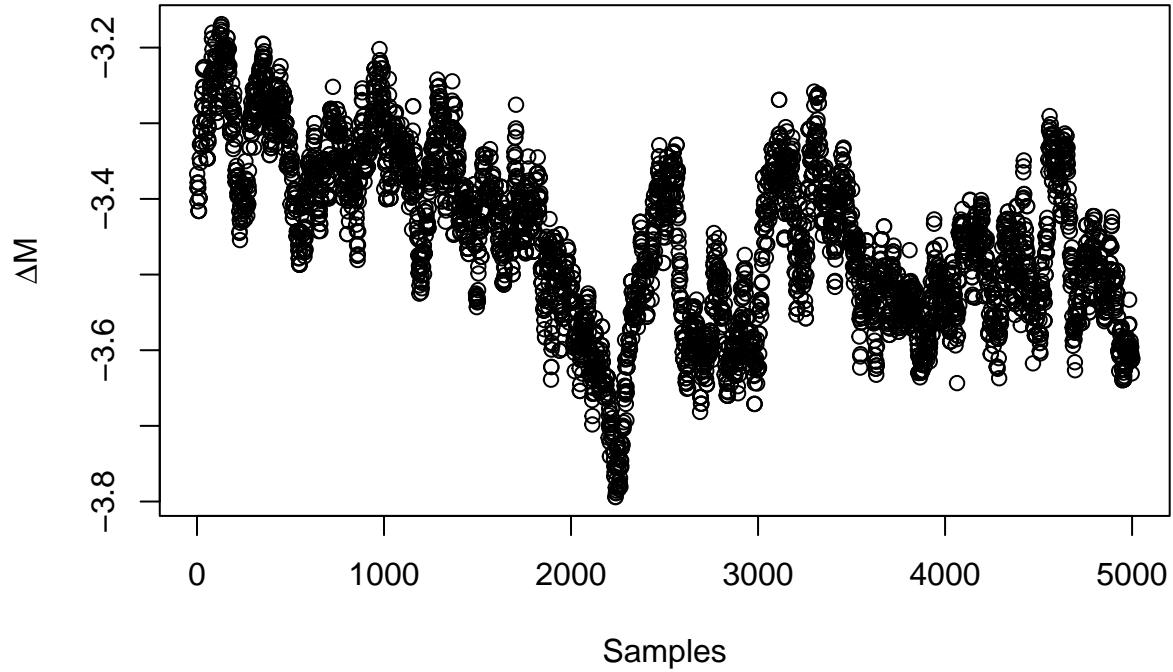
```
plot(mutationTrace$mix1$TTC, main = "Mixture 1: TTC", ylab = expression(Delta * M), xlab = "Samples")
```

### Mixture 1: TTC



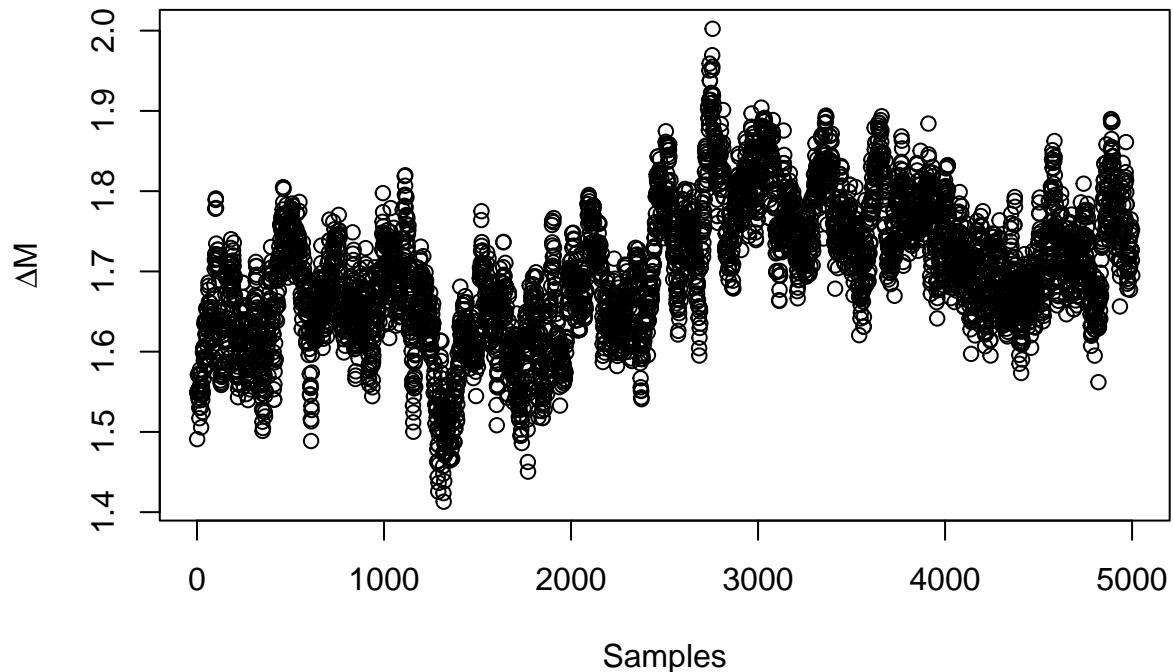
```
plot(mutationTrace$mix2$AGA, main = "Mixture 2: AGA", ylab = expression(Delta * M), xlab = "Samples")
```

**Mixture 2: AGA**



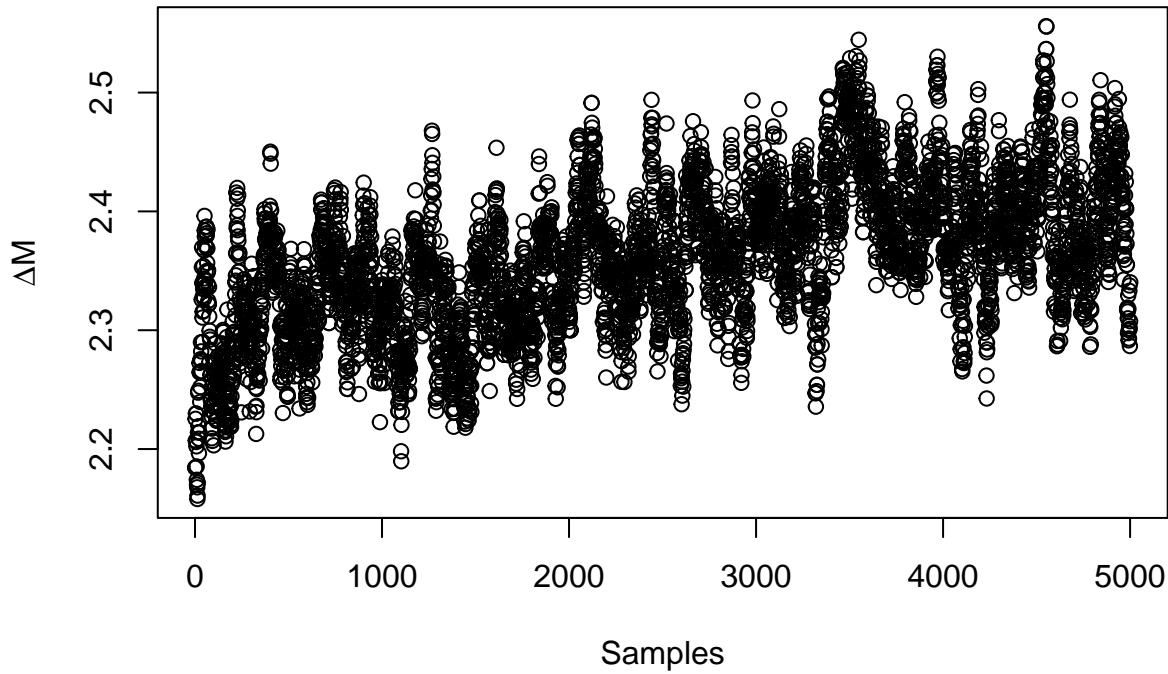
```
plot(mutationTrace$mix2$AGC, main = "Mixture 2: AGC", ylab = expression(Delta * M), xlab = "Samples")
```

**Mixture 2: AGC**



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

## Mixture 2: TTC



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

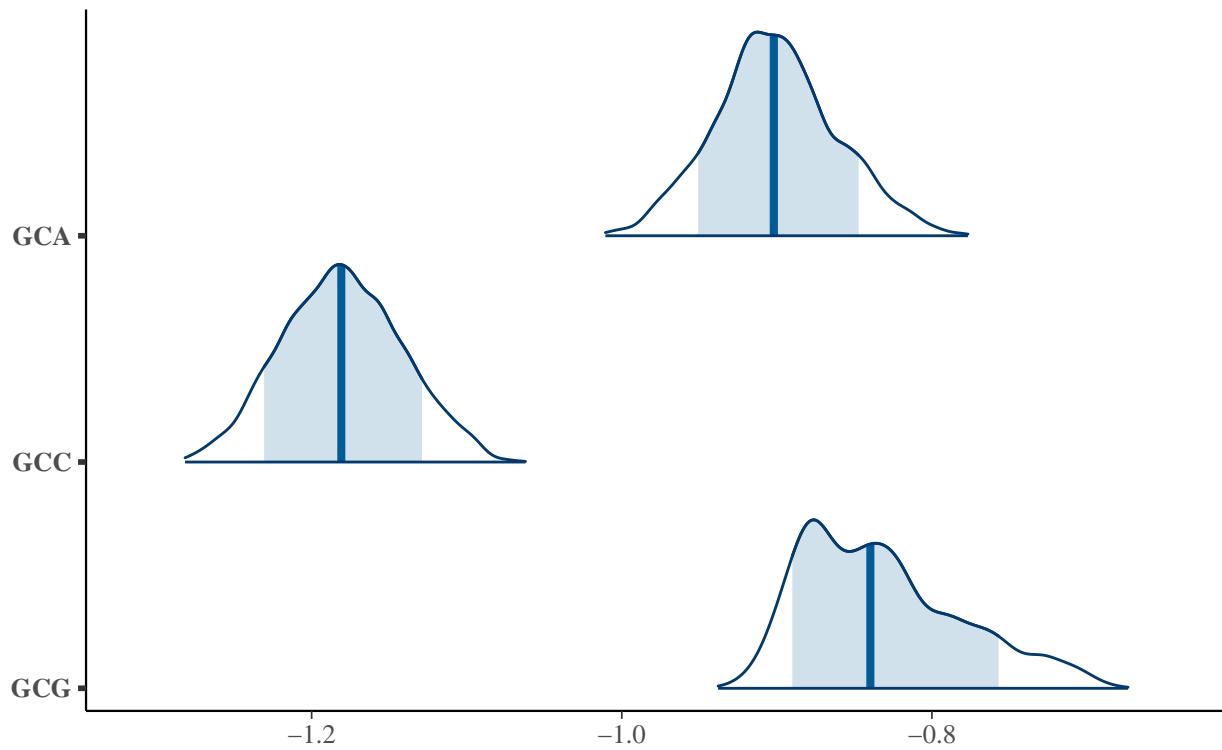
```
#Making trace data frames and discard burn-in
mutTrace.1 <- mutationTrace$mix1 %>% data.frame() #>% slice(-1:-2000)
mutTrace.2 <- mutationTrace$mix2 %>% data.frame() #>% slice(-1:-2000)

#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,
           pars = c("GCA", "GCC", "GCG"),
           prob = 0.8) +
  ggtitle("Posterior distributions: Codons of Alanine [Mixture 1]", #supply title
          "with medians and 80% intervals")# +
```

## Posterior distributions: Codons of Alanine [Mixture 1]

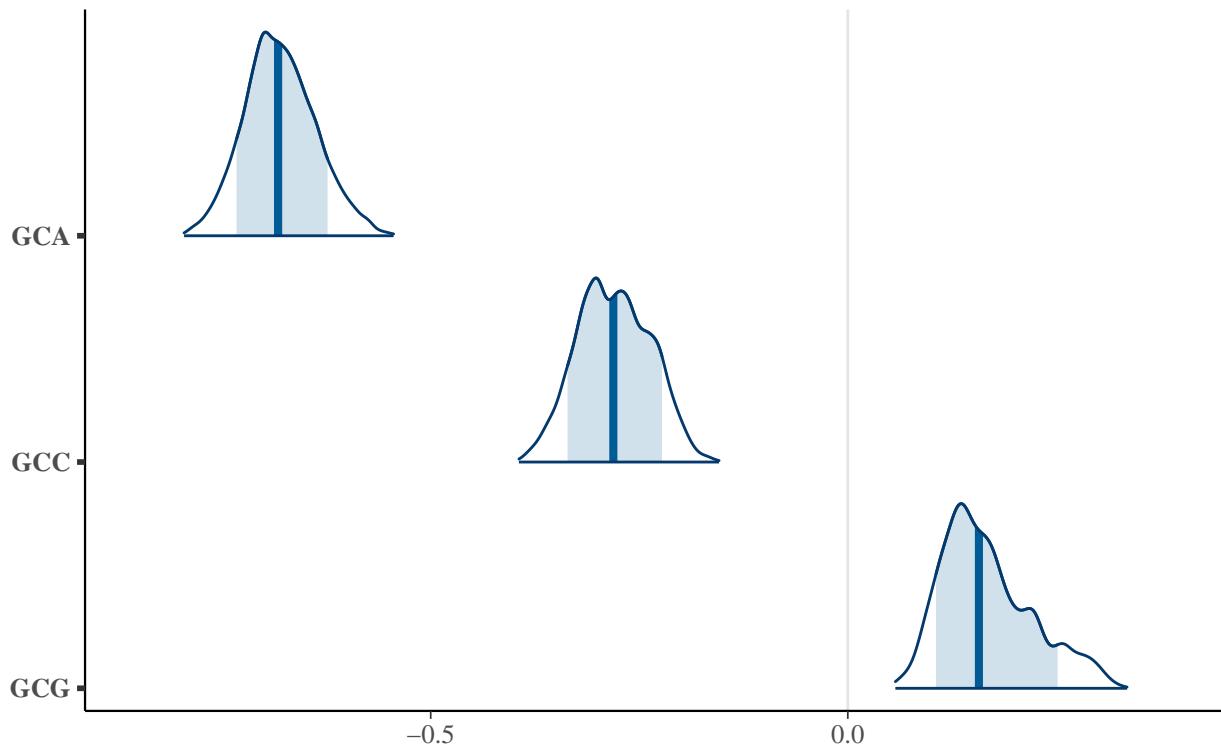
with medians and 80% intervals



```
#coord_cartesian(xlim = c(-1.5, 0.5)) #adjust x-axis
mcmc_areas(posterior.2, #supply the posterior matrix
            pars = c("GCA", "GCC", "GCG"),
            prob = 0.8) + #supply the desired traces to plot
#supply desired interval
ggttitle("Posterior distributions: Codons of Alanine [Mixture 2]", #supply title
         "with medians and 80% intervals")# +
```

## Posterior distributions: Codons of Alanine [Mixture 2]

with medians and 80% intervals



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

We can also directly compare codons across mixtures

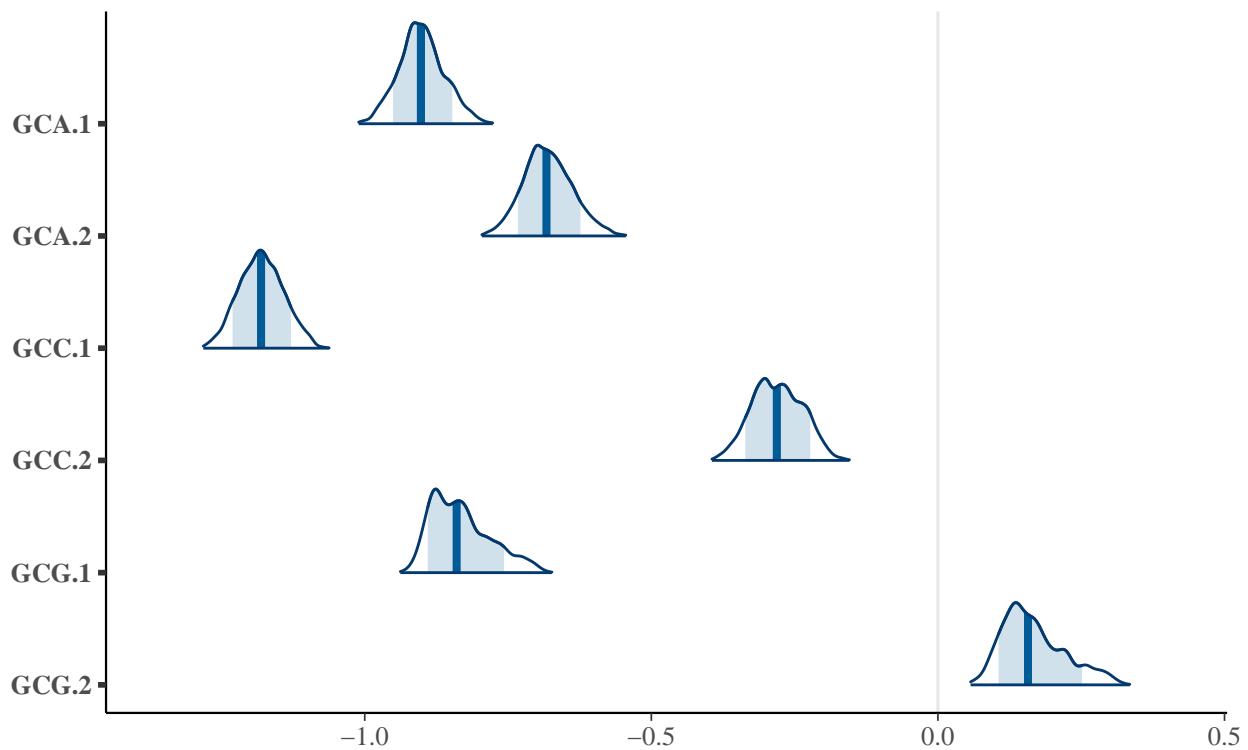
```
#Example: show all codons of alanine across the mixtures
alanine <- data.frame(
  GCA.1 = mutTrace.1$GCA,
  GCA.2 = mutTrace.2$GCA,
  GCC.1 = mutTrace.1$GCC,
  GCC.2 = mutTrace.2$GCC,
  GCG.1 = mutTrace.1$GCG,
  GCG.2 = mutTrace.2$GCG
)

posterior <- as.matrix(alanine)

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

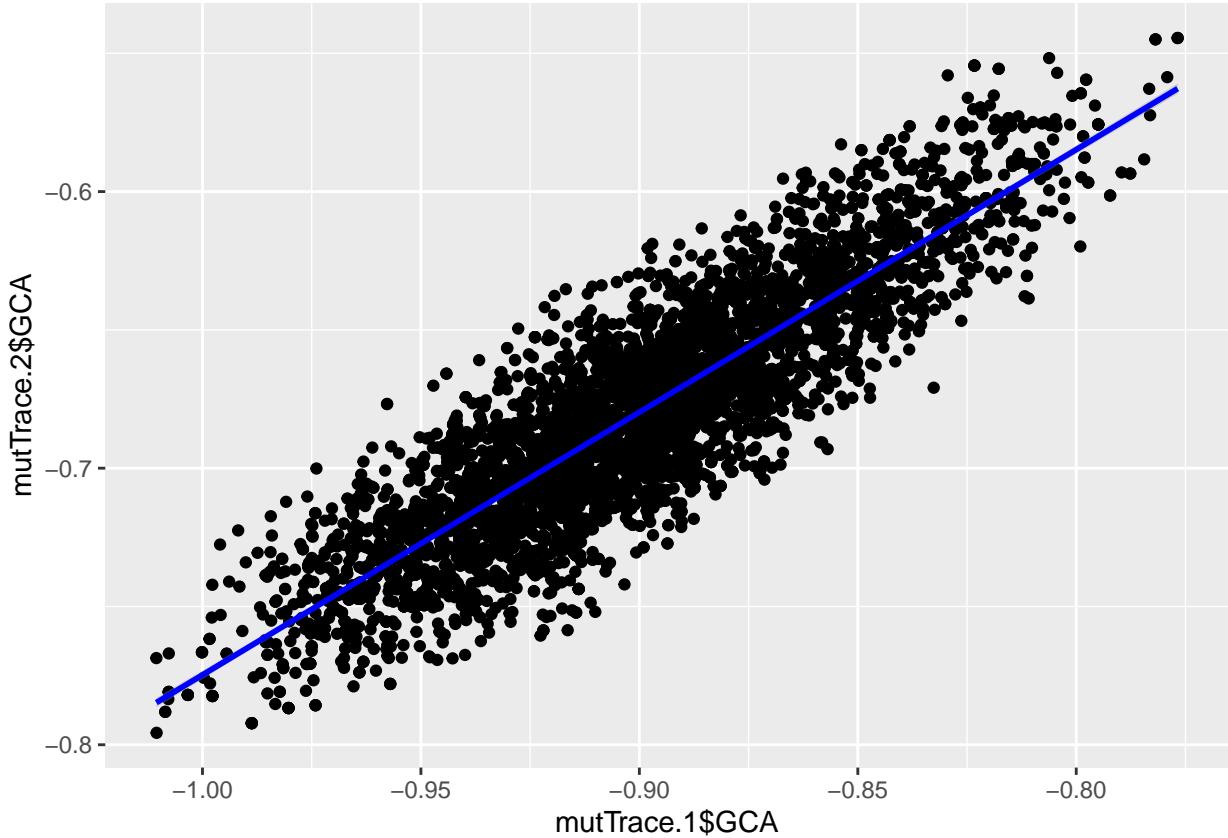
## Alanine Posterior Distributions

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'
```



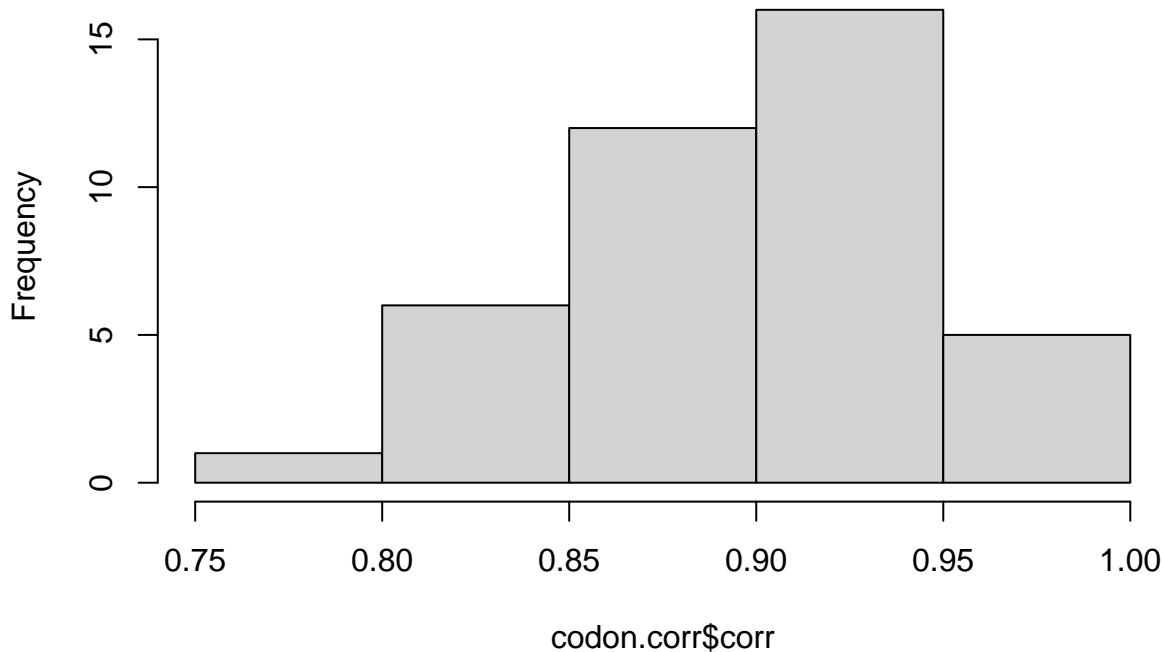
```
regression <- lm(mutTrace.2$GCA ~ mutTrace.1$GCA)
summary(regression)
```

```
##
## Call:
## lm(formula = mutTrace.2$GCA ~ mutTrace.1$GCA)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.063224 -0.013638  0.000022  0.013625  0.060522
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.174914  0.006543  26.73  <2e-16 ***
## mutTrace.1$GCA 0.949660  0.007259 130.83  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0202 on 4999 degrees of freedom
## Multiple R-squared:  0.774, Adjusted R-squared:  0.7739
## F-statistic: 1.712e+04 on 1 and 4999 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]])
})
```

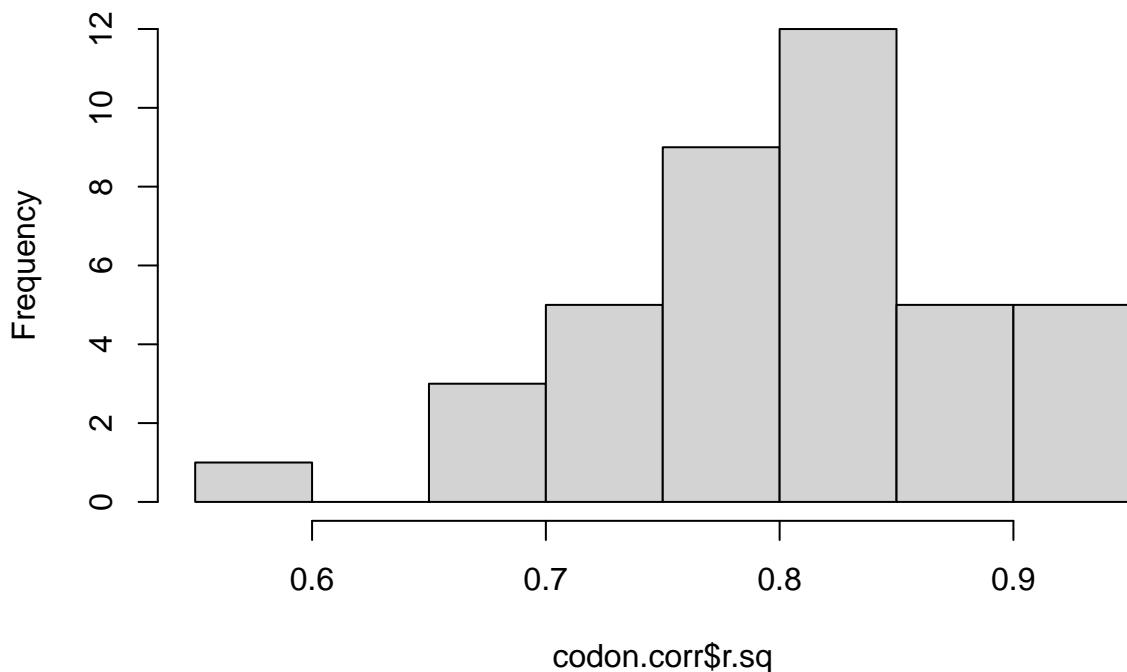
```
codon.corr <- data.frame(  
  codon = names(correlations),  
  corr = correlations,  
  r.sq = correlations^2  
)  
  
hist(codon.corr$corr)
```

Histogram of codon.corr\$corr



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

### Histogram of codon.corr\$r.sq



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

```
##      codon      corr      r.sq
##  GCA      GCA 0.8797516 0.7739629
##  GCC      GCC 0.8617701 0.7426477
##  GCG      GCG 0.9297271 0.8643925
##  TGC      TGC 0.8772133 0.7695032
##  GAC      GAC 0.9152369 0.8376585
##  GAA      GAA 0.8674361 0.7524454
##  TTC      TTC 0.9301517 0.8651822
##  GGA      GGA 0.9539228 0.9099687
##  GGC      GGC 0.9003157 0.8105683
##  GGG      GGG 0.8963725 0.8034837
##  CAC      CAC 0.9102120 0.8284858
##  ATA      ATA 0.9623069 0.9260346
##  ATC      ATC 0.9401958 0.8839681
##  AAA      AAA 0.8741927 0.7642129
##  CTA      CTA 0.8871385 0.7870147
##  CTC      CTC 0.8285537 0.6865012
##  CTG      CTG 0.9566770 0.9152308
##  CTT      CTT 0.8109166 0.6575858
##  TTA      TTA 0.8375679 0.7015199
##  AAC      AAC 0.9537579 0.9096542
##  CCA      CCA 0.9129154 0.8334145
##  CCC      CCC 0.9089776 0.8262402
##  CCG      CCG 0.9624586 0.9263266
##  CAA      CAA 0.9275190 0.8602916
##  AGA      AGA 0.9135368 0.8345495
##  AGG      AGG 0.8172028 0.6678203
```

```

## CGA    CGA 0.9140222 0.8354366
## CGC    CGC 0.9018301 0.8132975
## CGG    CGG 0.7564980 0.5722892
## TCA    TCA 0.8868092 0.7864306
## TCC    TCC 0.8728411 0.7618515
## TCG    TCG 0.8512788 0.7246755
## ACA    ACA 0.8666557 0.7510921
## ACC    ACC 0.9168359 0.8405881
## ACG    ACG 0.8463827 0.7163636
## GTA    GTA 0.9076406 0.8238115
## GTC    GTC 0.8477263 0.7186399
## GTG    GTG 0.8803802 0.7750693
## TAC    TAC 0.9108389 0.8296274
## AGC    AGC 0.9433628 0.8899333

# 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 0.9496599 0.007258657 130.83134     0
## GCC    GCC 0.9733265 0.008103816 120.10718     0
## GCG    GCG 0.9811500 0.005496425 178.50696     0
## TGC    TGC 0.8613132 0.006667251 129.18565     0
## GAC    GAC 0.9265997 0.005769416 160.60545     0
## GAA    GAA 0.9354609 0.007588956 123.26608     0
## TTC    TTC 0.8812305 0.004920030 179.11079     0
## GGA    GGA 0.9062003 0.004031497 224.78013     0
## GGC    GGC 1.0231665 0.006995781 146.25478     0
## GGG    GGG 0.8921377 0.006240237 142.96537     0
## CAC    CAC 0.9091072 0.005850339 155.39396     0
## ATA    ATA 0.8571973 0.003426415 250.17322     0
## ATC    ATC 0.9240348 0.004734970 195.15114     0
## AAA    AAA 0.8949863 0.007031171 127.28837     0
## CTA    CTA 0.7988445 0.005877655 135.91210     0
## CTC    CTC 0.8670050 0.008286618 104.62711     0
## CTG    CTG 0.9598253 0.004131467 232.32072     0
## CTT    CTT 0.7410962 0.007563672  97.98101     0
## TTA    TTA 0.7424819 0.006849864 108.39366     0
## AAC    AAC 0.8853084 0.003946108 224.34976     0
## CCA    CCA 0.8960757 0.005666192 158.14427     0

```

## CCC	CCC	0.8426984	0.005465779	154.17718	0
## CCG	CCG	0.9563899	0.003814755	250.70807	0
## CAA	CAA	0.8889851	0.005066889	175.44988	0
## AGA	AGA	0.8245032	0.005192289	158.79379	0
## AGG	AGG	0.6486857	0.006470673	100.25011	0
## CGA	CGA	0.8332865	0.005230734	159.30585	0
## CGC	CGC	1.1163691	0.007565133	147.56768	0
## CGG	CGG	0.7108140	0.008691241	81.78510	0
## TCA	TCA	0.8488986	0.006256819	135.67576	0
## TCC	TCC	0.9213320	0.007285567	126.45990	0
## TCG	TCG	0.8545310	0.007449667	114.70727	0
## ACA	ACA	0.8258818	0.006724331	122.81991	0
## ACC	ACC	0.9678598	0.005961282	162.35767	0
## ACG	ACG	0.9053248	0.008057068	112.36404	0
## GTA	GTA	0.9714352	0.006353998	152.88566	0
## GTC	GTC	0.9462949	0.008374532	112.99676	0
## GTG	GTG	0.9780576	0.007452070	131.24644	0
## TAC	TAC	0.8813103	0.005648661	156.02108	0
## AGC	AGC	0.8868789	0.004411355	201.04454	0