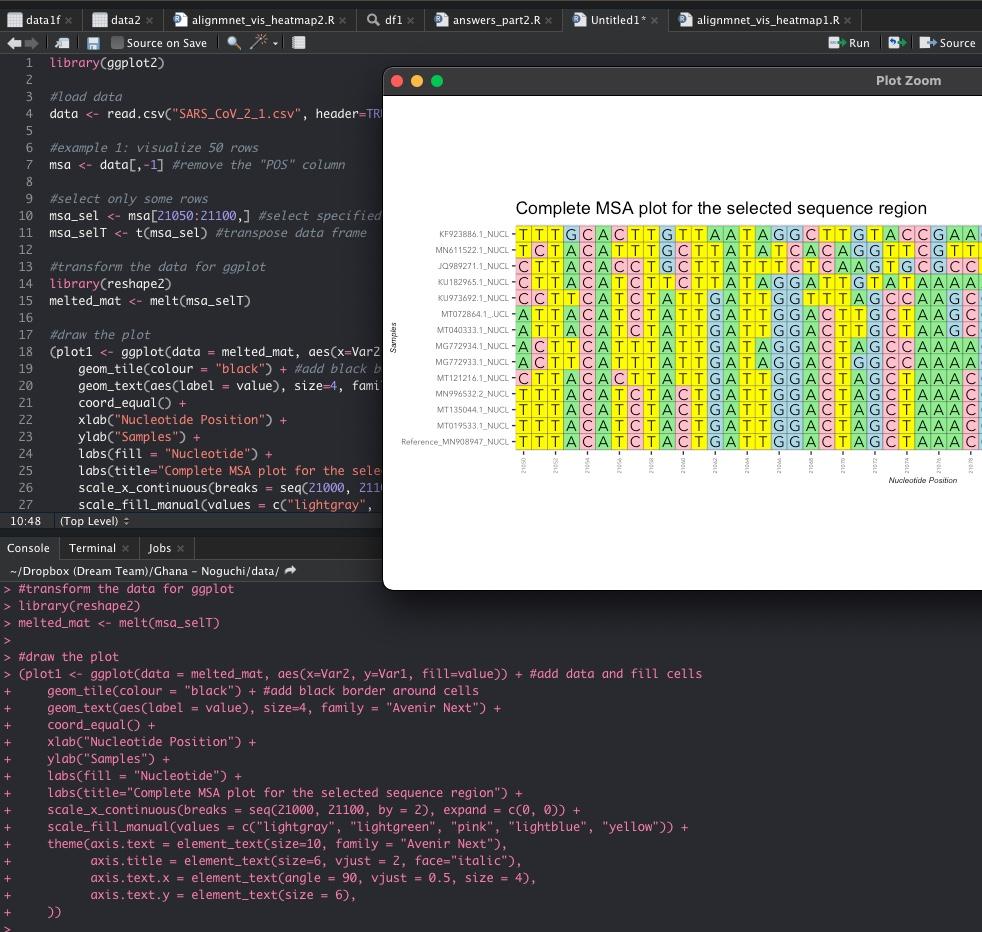
# Multiple Sequence Alignment - visualizing and filtering alignment in R



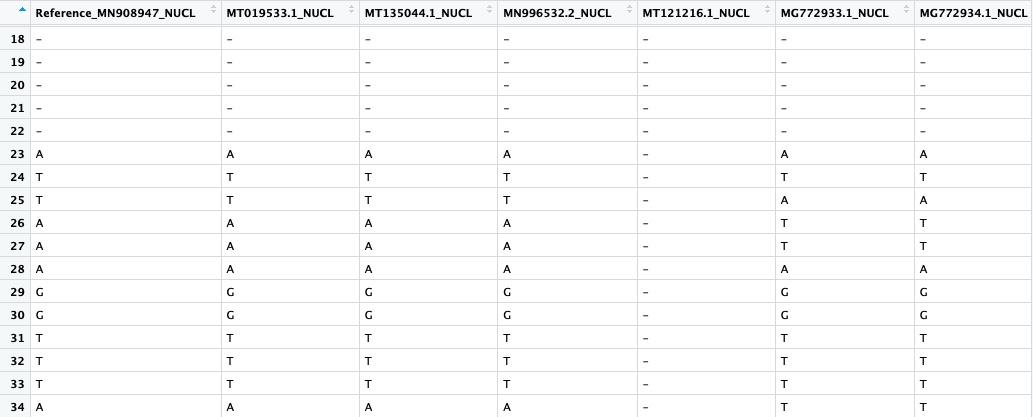
## Outline:

1. The challenge
2. Preparing Data
3. Making a Heatmap using the heatmap.2 package
4. Making a Heatmap in ggplot2
5. Filtering data to create a report
6. Considering the type of variation
7. Assignment

## 1. The challenge

Many times, multiple sequence alignment (MSA) gives us a file with sequence alignment of several thousand nucleotides. Not only are these files difficult to navigate effectively, they also require specialized software to make them colorful, which in turn makes differences easy to spot. In this tutorial, we will take an output from the T-Bioinfo platform after MSA and learn how to visualize and filter the table to create a report on important variants in specific positions of the reference genome.

## 2. Preparing the data



The input data contains sequence alignment in a CSV file. The table includes positions and 14 genomes from various coronaviruses. To generate this file, we ran an MSA pipeline on server.t-bio.info. For the purpose of this tutorial, we used a file on our GitHub repository: <https://raw.githubusercontent.com/PineBiotech/omicslogic/master/SARS_CoV_2_1.csv>

To load the file, we have to load the table and assign it to a dataframe.

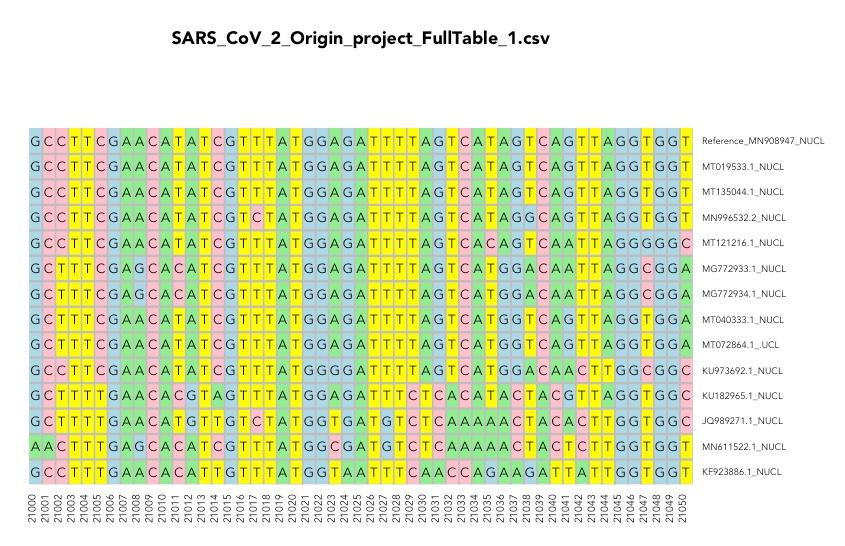
#load and prepare data

msa1 <- read.csv("SARS\_CoV\_2\_1.csv", header=TRUE)

## 3. Making a Heatmap using the heatmap.2 package

### Preparing the Data:

In this first step of the tutorial, we will use the heatmap.2 package which relies on the base plotting functionality of R to make a visual alignment plot:



First, we need to load the library “gplots that contains a function “heatmap.2” for plotting the heatmap:

library(gplots) #library that contains functions for drawing plots like heatmap.2

Then, we can load the data:

#load and prepare data

msa1 <- read.csv("https://raw.githubusercontent.com/PineBiotech/omicslogic/master/SARS\_CoV\_2\_1.csv", header=TRUE)

### Example 1:

In our first example, let’s just visualize a certain small portion of the genome alignment selecting 50 nucleotide positions:

#example 1: visualize sample dataset (some rows)

msa1m <- data.matrix(msa1) #prepare a numeric matrix for msa1 data frame

msa1 <- msa1[,-1] #remove the "POS" column

#select only some rows

msa1sel <- msa1[21000:21050,] #select from the letter data frame

msa1msel <- msa1m[21000:21050,] #select from the numeric matrix

row.names(msa1msel) <- msa1msel[,1] #for numeric matrix, make sure to add the position to row names

msa1msel <- msa1msel[,-1] #now, remove the POS column from the input data

#transpose

msa1selT <- t(msa1sel)

msa1mselT <- t(msa1msel)

Now, we can make a theme for our plot by setting colors and margins. Since we expect to take a portion of the data, we do not know if 4 or 5 variations will be included, so we will check whether the data we selected contains the “-” character or not, and then decide how many colors to set:

#set colors and margins for plot

if("-" %in% msa1selT){ #first, check if we have 4 or 5 characters (I am assuming that TCGA will be present, so only looking for "-")

TCGAcolors <- c("white", "lightgreen", "pink", "lightblue", "yellow")

names(TCGAcolors) = c(1,2,3,4,5) #labels = c("-","A","T","C","G")

} else {TCGAcolors <- c("lightgreen", "pink", "lightblue", "yellow") }

par(cex.main=0.9, family="avenir") #set plot margins

Then, we can use the heatmap.2 function to make our plot:

#draw heat map for first selected positions of alignment

heatmap.2(msa1mselT, #data source matrix

#main settings

cexRow = 0.7, #row name font size

col = TCGAcolors, #set colors

dendrogram = "none", #remove dendrogram

Rowv = FALSE, #no reordering for rows

Colv = FALSE, #no reordering for columns

density.info="none", #remove density info

trace="none", #remove row and column lines

offsetRow=0.1, #change position of the row names

offsetCol=0.1, #change position of the column names

#add gray borders between cells

sepwidth=c(0.05,0.05), #sets separation width and height

sepcolor="gray", #color for border

colsep=1:ncol(msa1mselT), #add separation for number of columns in source data

rowsep=1:nrow(msa1mselT), #add separation for number of rows in source data

#plot title

main = "SARS\_CoV\_2\_Origin\_project\_FullTable\_1.csv", #heat map title

#plot margins

margins = c(5,10), #set margins

lwid=c(0.2,4),

lhei=c(0.9,3),

#adding letters inside the heatmap

notecex=1.0, #size of font inside each cell

cellnote = msa1selT, #data to use in cells

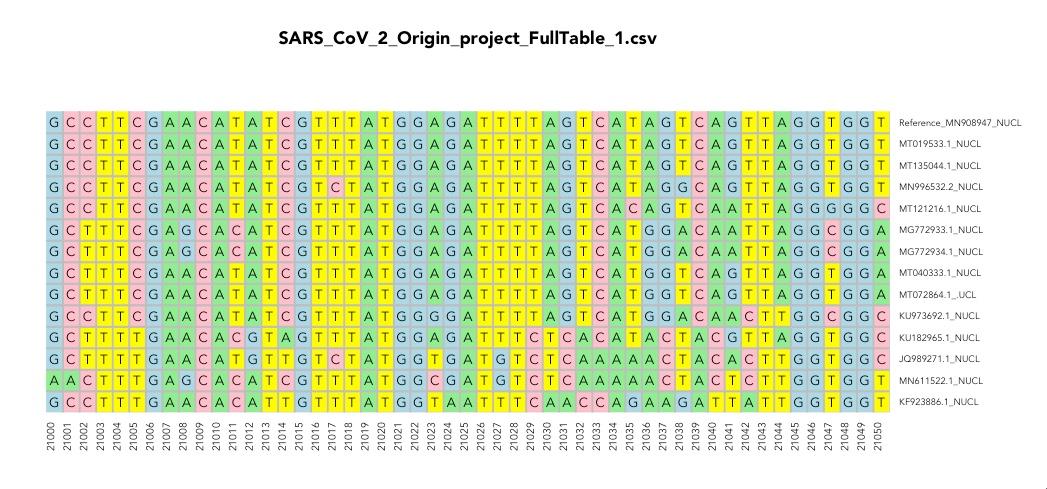
notecol="black", #font color for cells

#legend

key = FALSE

)

Result:



### Example 2:

In the second example, we will take the selected region and prepare a report on only those positions that contain variation from any of the samples as compared to the reference genome:

#select only data where any sample has a variant to create a report

new\_df <- msa1sel[!(msa1sel[2] == msa1sel[3] & msa1sel[2] == msa1sel[4] & msa1sel[2] == msa1sel[5] & msa1sel[2] == msa1sel[6] & msa1sel[2] == msa1sel[7] & msa1sel[2] == msa1sel[8] & msa1sel[2] == msa1sel[9] & msa1sel[2] == msa1sel[10] & msa1sel[2] == msa1sel[11] & msa1sel[2] == msa1sel[12] & msa1sel[2] == msa1sel[13] & msa1sel[2] == msa1sel[14]),]

new\_dfT <- t(new\_df)

new\_matrix <- msa1msel[!(msa1msel[,2] == msa1msel[,3] & msa1msel[,2] == msa1msel[,4] & msa1msel[,2] == msa1msel[,5] & msa1msel[,2] == msa1msel[,6] & msa1msel[,2] == msa1msel[,7] & msa1msel[,2] == msa1msel[,8] & msa1msel[,2] == msa1msel[,9] & msa1msel[,2] == msa1msel[,10] & msa1msel[,2] == msa1msel[,11] & msa1msel[,2] == msa1msel[,12] & msa1msel[,2] == msa1msel[,13] & msa1msel[,2] == msa1msel[,14]),]

new\_matrixT <- t(new\_matrix)

#draw heat map for only variants

heatmap.2(new\_matrixT, #data source

#main settings

cexRow = 0.7, #row name font size

col = TCGAcolors, #set colors

dendrogram = "none", #remove dendrogram

Rowv = FALSE, #no reordering for rows

Colv = FALSE, #no reordering for columns

density.info="none", #remove density info

trace="none", #remove row and column lines

offsetRow=0.1, #change position of the row names

offsetCol=0.1, #change position of the column names

#add gray borders between cells

sepwidth=c(0.05,0.05), #sets separation width and height

sepcolor="gray", #color for border

colsep=1:ncol(new\_matrixT), #add separation for number of columns in source data

rowsep=1:nrow(new\_matrixT), #add separation for number of rows in source data

#plot title

main = cbind("Number of Variants Found: ",ncol(new\_matrixT)), #heat map title

#plot margins

margins = c(5,10), #set margins

lwid=c(0.2,4),

lhei=c(0.9,3),

#adding letters inside the heatmap

notecex=1.0, #size of font inside each cell

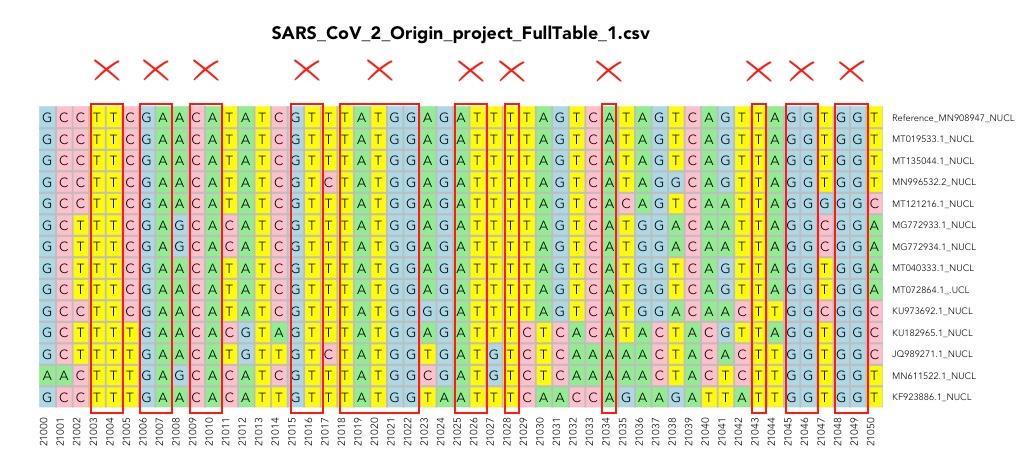
cellnote = new\_dfT, #data to use in cells

notecol="black", #font color for cells

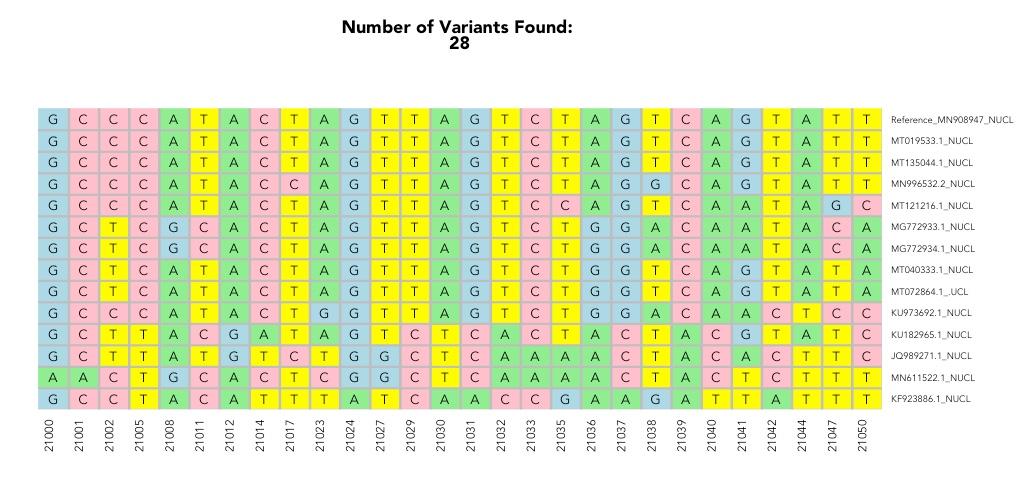
#legend

key = FALSE

)



Result:



As you can see, the alignment contains positions 2100-21005, then skips positions 21006-21007, 21009-21010 and a few others. Those were filtered out because they were matching across all samples. As a result, we find only 28 variants in this 50-NT long sequence.

### ASSIGNMENT:

Find variants in the E protein (E 26245-26472 Length: 227NT)

## 4. Making a Heatmap using ggplot2

One of the challenges in a regular plot is the use of numeric data for the actual graphic. As a result, we have to perform additional steps before creating the plot. Regular plots also have limited control over the look of the plot. Ggplot2 is a library dedicated to addressing these issues in R, so let’s see how the same data can be visualized using ggplot2.

First, we need to call the library:

library(ggplot2)

Then, load the data

#load data

data <- read.csv("SARS\_CoV\_2\_1.csv", header=TRUE)

Then, load the data

#example 1: visualize 50 rows

msa <- data[,-1] #remove the "POS" column

#select only some rows

msa\_sel <- msa[21050:21100,] #select specific rows from the letter data frame

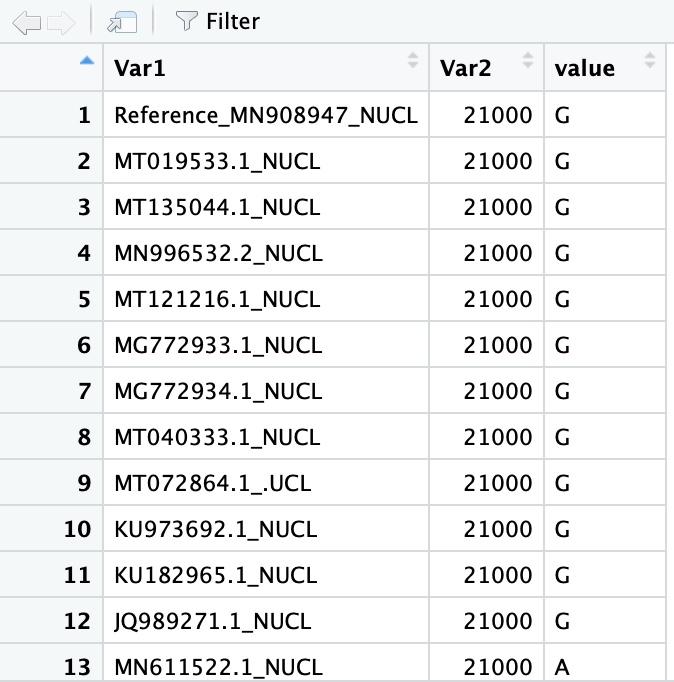
msa\_selT <- t(msa\_sel) #transpose data frame

One of the main things we have to do when using ggplot2 is to transform the data into a “long format”. This can be done with the help of a library called “reshape2”:

#transform the data for ggplot

library(reshape2)

melted\_mat <- melt(msa\_selT)



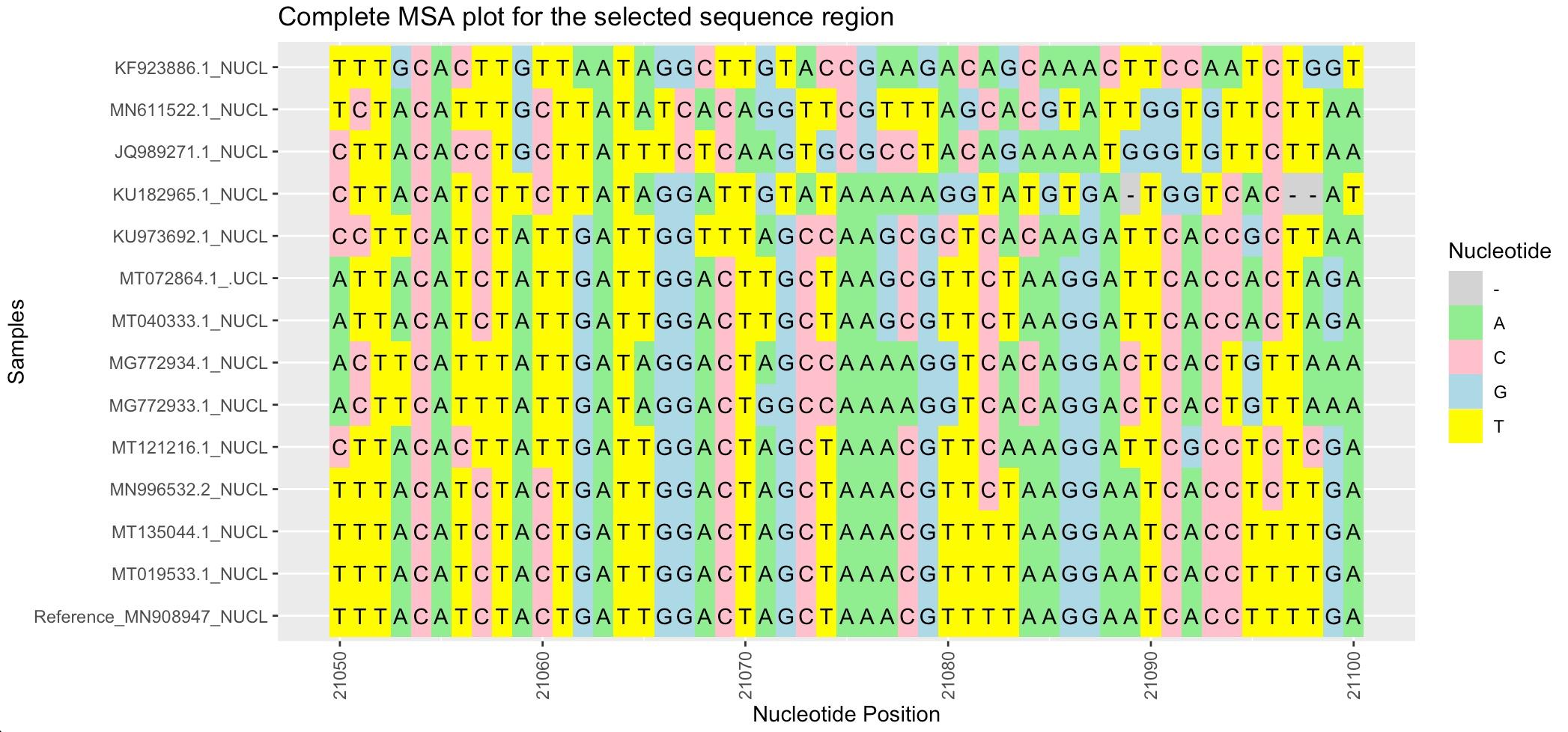
Now we can draw the plot itself:

#draw the plot

plot1 <- ggplot(data = melted\_mat, aes(x=Var2, y=Var1, fill=value)) + geom\_tile() + geom\_text(aes(label = value)) + xlab("Nucleotide Position") + ylab("Samples" ) + labs(fill = "Nucleotide") + labs(title="Complete MSA plot for the selected sequence region") + scale\_fill\_manual(values=c("lightgray", "lightgreen", "pink", "lightblue", "yellow")) + theme(axis.text.x = element\_text(angle = 90, vjust = 0.5))

plot(plot1)

This is the result:



This plot can be improved, however with some additional styling. We will also add parenthesis around the plot object to make our code styling easier to read:

(plot1 <- ggplot(data = melted\_mat, aes(x=Var2, y=Var1, fill=value)) + #add data and fill cells

geom\_tile(colour = "black") + #add black border around cells

geom\_text(aes(label = value), size=4, family = "Avenir Next") + #change text settings for text inside the cells

coord\_equal() + #this controls the cell proportion

xlab("Nucleotide Position") + #adds label to X axis

ylab("Samples") + #adds label to Y axis

labs(fill = "Nucleotide") + #Title for legend

labs(title="Complete MSA plot for the selected sequence region") + #main title

scale\_x\_continuous(breaks = seq(21000, 21100, by = 2), expand = c(0, 0)) + #adds breaks every 2 positions

scale\_fill\_manual(values = c("lightgray", "lightgreen", "pink", "lightblue", "yellow")) + #adds color scheme

theme(axis.text = element\_text(size=10, family = "Avenir Next"),

axis.title = element\_text(size=6, vjust = 2, face="italic"),

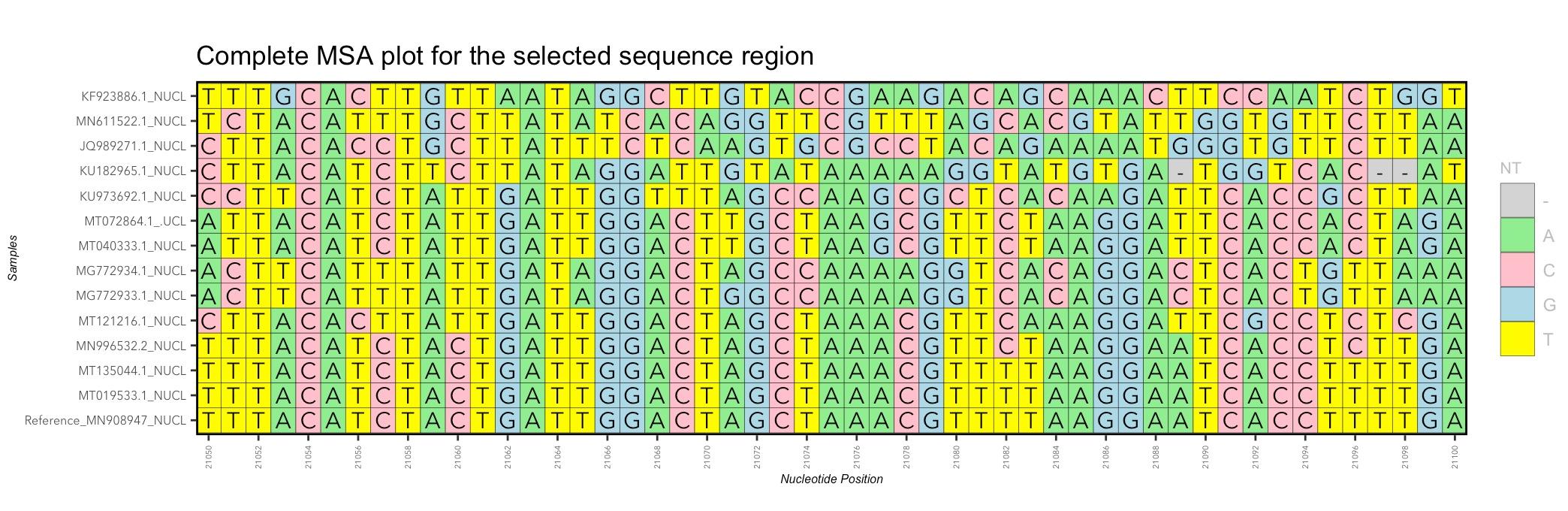
axis.text.x = element\_text(angle = 90, vjust = 0.5, size = 4),

axis.text.y = element\_text(size = 6),

))

plot(plot1)

And here is the result:



The problem here is that if the number of nucleotides (-,A,T,C,G) changes, the coloring scheme will also change. To address this issue, we need to specify what the colors are for each nucleotide, like this:

scale\_fill\_manual(values = c("-" = "lightgray", "A" = "lightgreen", "C" = "pink", "G" = "lightblue", "T" = "yellow", "V"="red", "\*"="white"))

## 5. Filtering data and making a variant report.

Now, we should learn how to mark variants only. To do that, we will use an “ifelse” statement before creating the object for visualization.

The filter can be done in the following way:

#select only data where any sample has a variant,

new\_df <- msa\_sel[!(msa\_sel[2] == msa\_sel[3] & msa\_sel[2] == msa\_sel[4] & msa\_sel[2] == msa\_sel[5] & msa\_sel[2] == msa\_sel[6]& msa\_sel[2] == msa\_sel[7]),]

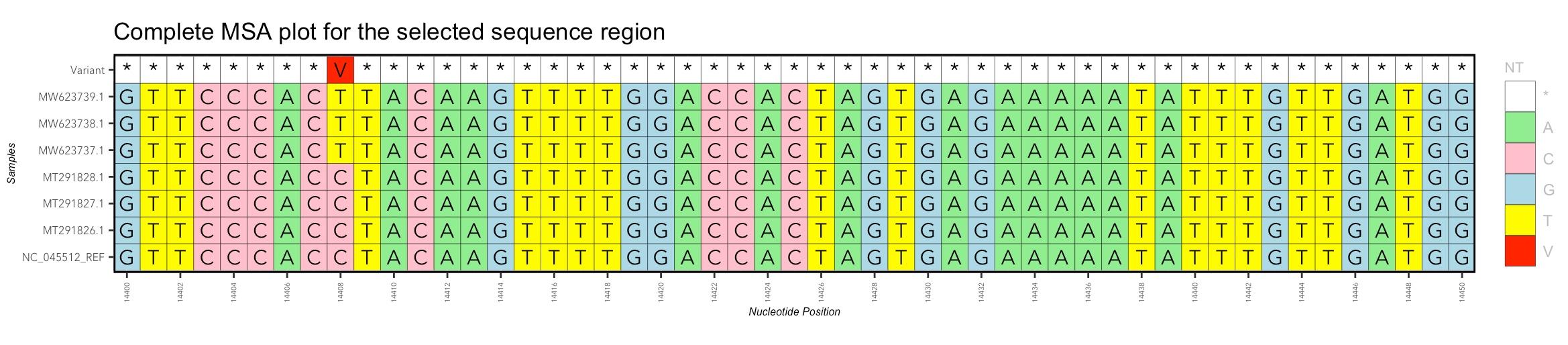
Here, we are eliminating the positions which are the same as the reference. But this removes the data which will cause an issue with visualization because of the number of positions that will be in the object itself, so instead, let’s add another column to the data which will mark the variants.

msa\_sel <- msa[startnt:endnt,] #select specified rows from the letter data frame

#select only data where any sample has a variant,

msa\_sel$Variant <- ifelse(msa\_sel[2] == msa\_sel[3] & msa\_sel[2] == msa\_sel[4] & msa\_sel[2] == msa\_sel[5] & msa\_sel[2] == msa\_sel[6]& msa\_sel[2] == msa\_sel[7], "\*", "V")

As a result, we will see a new column that looks like this:



### The Result

Final script

library(ggplot2)

library(gghighlight)

library(stringr)

#load data

msa <- read.csv("SARS\_CoV\_2\_1.csv", header=TRUE)

#example 1: visualize 50 rows

msa <- msa[,-1] #remove the "POS" column

msa1 <- msa[,-1] #remove the reference column

startnt <- 600

endnt <- 650

msa\_sel <- msa[startnt:endnt,] #select specified rows from the letter data frame

#select only data where any sample has a variant,

msa\_sel$Variant <- ifelse(msa\_sel[2] == msa\_sel[3] & msa\_sel[2] == msa\_sel[4] & msa\_sel[2] == msa\_sel[5] & msa\_sel[2] == msa\_sel[6]& msa\_sel[2] == msa\_sel[7], "\*", "V")

msa\_selT <- t(msa\_sel) #transpose data frame

msa\_sel1 <- msa1[startnt:endnt,] #select specified rows from the letter data frame

msa\_sel1T <- t(msa\_sel1) #transpose data frame

#transform the data for ggplot

library(reshape2)

melted\_mat <- melt(msa\_selT)

melted\_mat1 <- melt(msa\_sel1T)

#draw the plot

(plot1 <- ggplot(melted\_mat) + #add data and fill cells

geom\_tile(data = melted\_mat, aes(x=Var2, y=Var1, fill=value), colour = "black") + #add black border around cells

geom\_text(data = melted\_mat, aes(x=Var2, y=Var1,label = value), size=4, family = "Avenir Next") +

scale\_fill\_manual(values = c("-" = "lightgray", "A" = "lightgreen", "C" = "pink", "G" = "lightblue", "T" = "yellow", "V"="red", "\*"="white")) +

coord\_equal() +

xlab("Nucleotide Position") +

ylab("Samples") +

labs(fill = "NT") +

labs(title="Complete MSA plot for the selected sequence region") +

scale\_x\_continuous(breaks = seq(startnt, endnt, by = 2), expand = c(0, 0)) +

theme(axis.text = element\_text(size=10, family = "Avenir Next"),

legend.title = element\_text(color = "gray", size = 8),

legend.text = element\_text(color = "gray"),

axis.title = element\_text(size=6, vjust = 2, face="italic"),

axis.text.x = element\_text(angle = 90, vjust = 0.5, size = 4),

axis.text.y = element\_text(size = 6),

panel.border = element\_rect(colour = "black", fill=NA, size=1)

))

#make the first plot

plot(plot1)

#Report on the number of mutations in each sample:

x <- ncol(msa\_sel)-1

#create an empty data frame to store the new data

daf2 <- data.frame(matrix(0, ncol = x+1, nrow = (endnt-startnt)+1))

colnames(daf2) <- colnames(msa\_sel)

i=2

for (i in 2:x) {

daf2[,i] <- ifelse(msa\_sel[,1]== msa\_sel[,i],0,1)

}

#Drop the variant column

daf3 <- subset(daf2, select = -Variant )

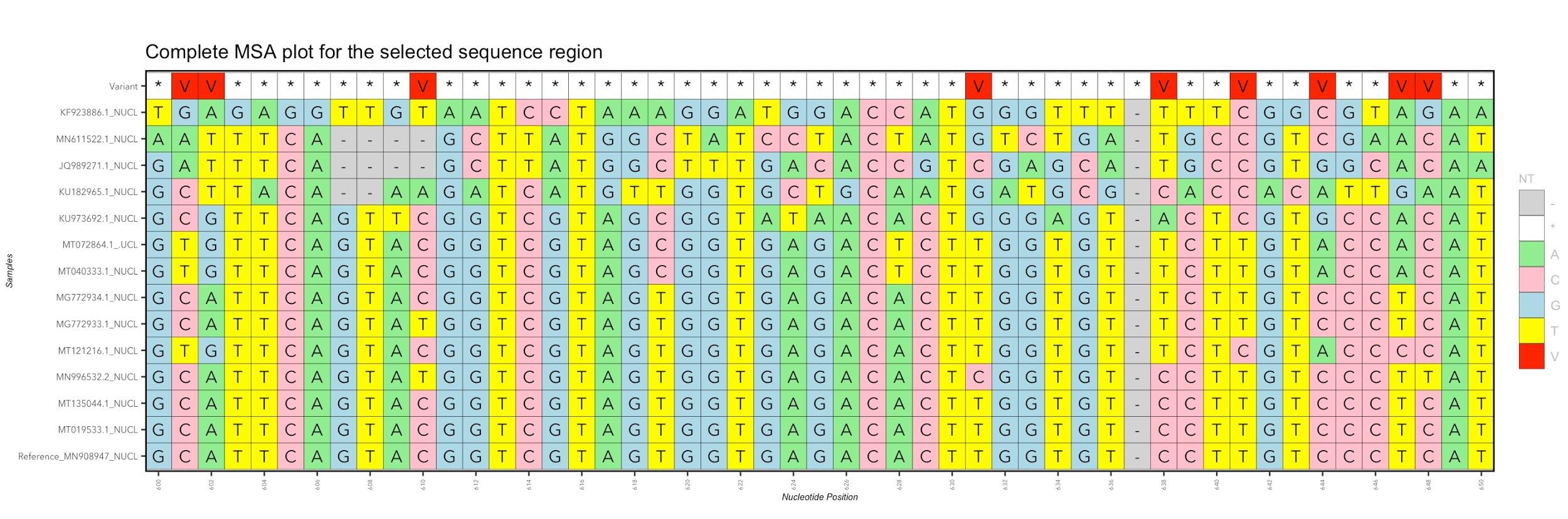
#count the number of mutations in each sample

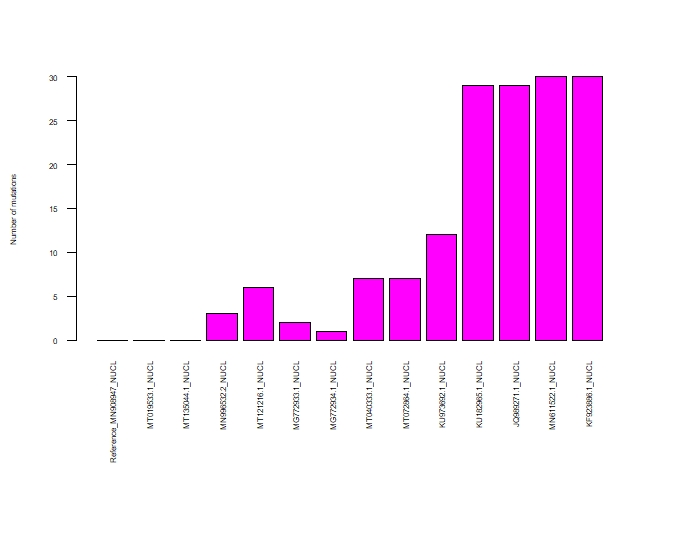
summ1 <- colSums(daf3)

#make a barplot

par(mar=c(11,4,4,4))

barplot(summ1, las=2, cex.axis = 0.5, cex.lab=0.5, cex.names=0.5, ylab="Number of mutations", col="magenta", space=0.2)

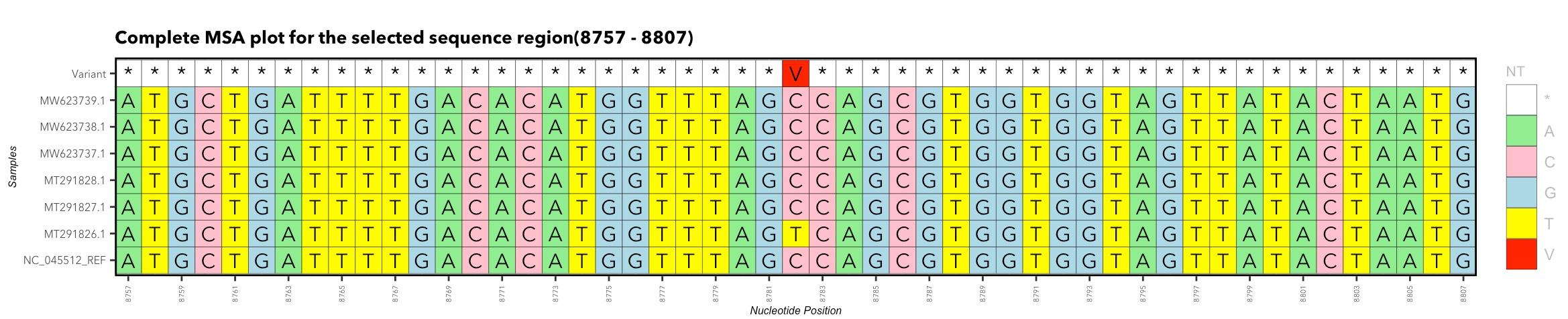
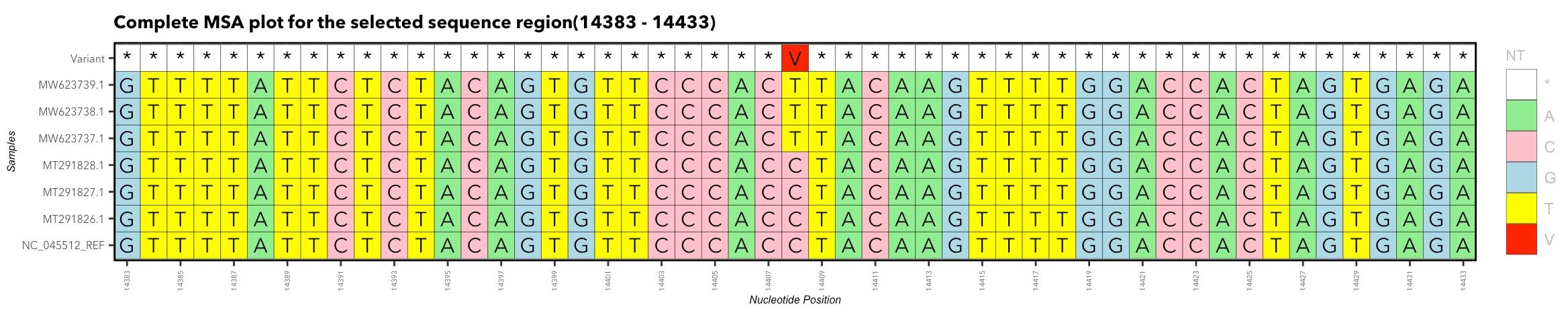




Let’s apply this data to a new dataset.

In this case, we will use SARS-COV-2 samples from early (2019) and late (2021) time points collected in China and the USA. We will then search for some common variants reporte, such as NT 8782 and 14408

|  |  |  |  |
| --- | --- | --- | --- |
| Clade | NT/AA | POS | Variant |
| 19A | nuc | 8782 | C |
| 19A | nuc | 14408 | C |



## 6. Making a function to build a report

To make sure we can prepare this report when we load the data, including both the section of interest where the variant was found and a barplot to summarize the data, we can create a function that takes in the positions of interest and does the rest automatically.

First, let’s make a list of positions we are interested in:

myPos <- list(1878,14408)

myPos[1] <- 8782

myPos[2] <- 14408

myPos[3] <- 28144

PosLength = length(myPos)

Then, we can make a new function called “generateReport”, which will take in the list of positions and prepare a report for each one:

generateReport <- function(myPos) {

This simply means that the new function will perform all of the plotting we did already, but for each item in the list “myPos”

Inside, place all of what we already prepared:

generateReport <- function(myPos) {

j = 1

for (j in 1:PosLength) {

#select 50 NT section around the variant of interest

startnt <- as.numeric(myPos[j])-25

endnt <- as.numeric(myPos[j])+25

#select only data where any sample has a variant,

msa\_sel <- msa[startnt:endnt,] #select specified rows from the letter data frame

msa\_sel[] <- lapply(msa\_sel, as.character) #if number of characters in each sample does not match, we can convert the values to characters

msa\_sel$Variant <- ifelse(msa\_sel[1] == msa\_sel[2] & msa\_sel[1] == msa\_sel[3] & msa\_sel[1] == msa\_sel[4] & msa\_sel[1] == msa\_sel[5] & msa\_sel[1] == msa\_sel[6], "\*", "V")

msa\_selT <- t(msa\_sel) #transpose data frame

msa\_sel1 <- msa1[startnt:endnt,] #select specified rows from the letter data frame

msa\_sel1T <- t(msa\_sel1) #transpose data frame

#transform the data for ggplot

library(reshape2)

melted\_mat <- melt(msa\_selT)

melted\_mat1 <- melt(msa\_sel1T)

#create a tile for the MSA plot

plottitle <- as.character(c("Complete MSA plot for the selected sequence region","(",startnt," - ",endnt,")"))

plottitle1 <- paste( unlist(plottitle), collapse='')

#draw the plot

(plot1 <- ggplot(melted\_mat) + #add data and fill cells

geom\_tile(data = melted\_mat, aes(x=Var2, y=Var1, fill=value), colour = "black") + #add black border around cells

geom\_text(data = melted\_mat, aes(x=Var2, y=Var1,label = value), size=4, family = "Avenir Next") +

scale\_fill\_manual(values = c("-" = "lightgray", "A" = "lightgreen", "C" = "pink", "G" = "lightblue", "T" = "yellow", "V"="red", "\*"="white")) +

coord\_equal() +

xlab("Nucleotide Position") +

ylab("Samples") +

labs(fill = "NT") +

labs(title=plottitle1, size=2) +

scale\_x\_continuous(breaks = seq(startnt, endnt, by = 2), expand = c(0, 0)) +

theme(axis.text = element\_text(size=10, family = "Avenir Next"),

plot.title = element\_text(size = 10, face = "bold", family = "Avenir Next"),

legend.title = element\_text(color = "gray", size = 8),

legend.text = element\_text(color = "gray"),

axis.title = element\_text(size=6, vjust = 2, face="italic"),

axis.text.x = element\_text(angle = 90, vjust = 0.5, size = 4),

axis.text.y = element\_text(size = 6),

panel.border = element\_rect(colour = "black", fill=NA, size=1)

))

#make the first plot

plot(plot1)

###second plot

#Report on the number of mutations in each sample:

x <- ncol(msa\_sel)-1

#create an empty data frame to store the new data

daf2 <- data.frame(matrix(0, ncol = x+1, nrow = (endnt-startnt)+1))

colnames(daf2) <- colnames(msa\_sel)

#create a title for the second plot

varType <- as.factor(msa\_sel[msa\_sel$Variant == "V",])

varType1 <- as.character(varType)

plot2title <- as.character(c("Variant at ",myPos[j], "( ", varType1, " )"))

plot2title1 <- paste( unlist(plot2title), collapse='')

i=2

for (i in 2:x) {

daf2[,i] <- ifelse(msa\_sel[,1]== msa\_sel[,i],0,1)

}

#Drop the variant column

daf3 <- subset(daf2, select = -Variant )

#count the number of mutations in each sample

summ1 <- colSums(daf3)

#make a barplot

par(mar=c(11,4,4,4))

barplot1 <- barplot(summ1, las=2, cex.axis = 0.5, cex.lab=0.5, cex.names=0.5, col="magenta", ylab="Number of mutations",space=0.2, main=plot2title1)

}

}

Now, we can call the function to run it

generateReport(myPos)

## Report for COVID-19 proteins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gene symbol | Start | Stop | Length |  |
| ORF1ab | 266 | 21555 | 21289 |  |
| S | 21563 | 25384 | 3821 |  |
| ORF3a | 25393 | 26220 | 827 |  |
| E | 26245 | 26472 | 227 |  |
| M | 26523 | 27191 | 668 |  |
| ORF6 | 27202 | 27387 | 185 |  |
| ORF7a | 27394 | 27759 | 365 |  |
| ORF7b | 27756 | 27887 | 131 |  |
| ORF8 | 27894 | 28259 | 365 |  |
| N | 28274 | 29533 | 1259 |  |
| ORF10 | 29558 | 29674 | 116 |  |

## Spike protein

### **Barplot for No. of mutations:**

Code: To create a barplot with no of mutations in each sample

#Load data

msa <- read.csv("https://raw.githubusercontent.com/pine-bio-support/COVID-19-origin/main/SARS\_CoV\_2\_1.csv", header=TRUE)

#Define selected positions (Here, we are defining position for spike protein)

startnt <- 21563

endnt <- 25384

#select specified rows from the letter data frame

msa\_sel1 <- msa[startnt:endnt,]

#Extract position column

pos <- msa\_sel1[1]

msa\_sel <- msa\_sel1[,-1] #remove the "POS" column

#Report on the number of mutations in each sample:

x <- ncol(msa\_sel)

#create an empty data frame to store the new data

daf2 <- data.frame(matrix(0, ncol = x+1, nrow = (endnt-startnt)+1))

colnames(daf2) <- colnames(msa\_sel)

#define i value

i=2

for (i in 2:x) {

daf2[,i] <- ifelse(msa\_sel[,1]== msa\_sel[,i],0,1)

}

#Load library

library(dplyr) #required for modification of data

#Remove reference genome column and last column (NA column) from the dataframe

daf3 <- select(daf2, -ncol(daf2))

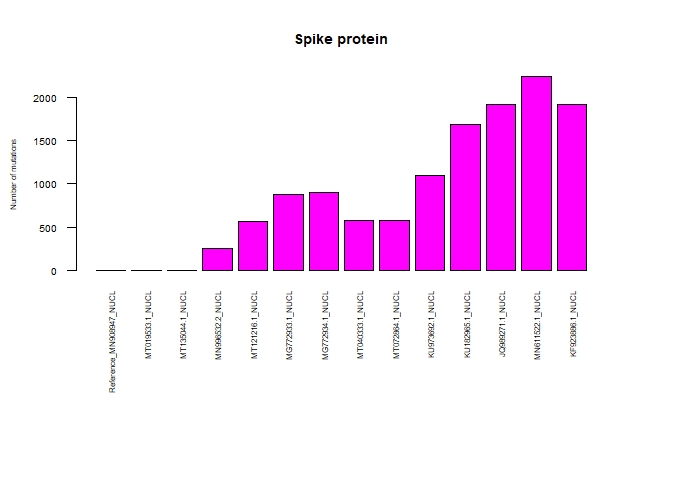
#count the number of mutations in each sample

summ1 <- colSums(daf3)

#make a barplot

par(mar=c(11,4,4,4))

barplot(summ1, las=2, main= "Spike protein", cex.axis = 0.6, cex.lab=0.5, cex.names=0.5, ylab="Number of mutations", col="magenta", space=0.2)



### **Table for type of variants**

Code: To create a Table with type of variants information

#Load data

msa <- read.csv("https://raw.githubusercontent.com/pine-bio-support/COVID-19-origin/main/SARS\_CoV\_2\_1.csv", header=TRUE)

#Define selected positions (Here, we are defining position for spike protein)

startnt <- 21563

endnt <- 25384

#select specified rows from the letter data frame

msa\_sel1 <- msa[startnt:endnt,]

#Extract position column

pos <- msa\_sel1[1]

msa\_sel <- msa\_sel1[,-1] #remove the "POS" column

#Report on the number of mutations in each sample:

x <- ncol(msa\_sel)

#create an empty data frame to store the new data

df\_new1 <- data.frame(matrix(0, ncol = x+1, nrow = (endnt-startnt)+1))

colnames(df\_new1) <- colnames(msa\_sel)

#define i value

i=2

#Create a for loop, to assign variant (i.e. nucleotide mutated to which new variant/nucleotide). Here, we assigning “-” if there is no mutation, else assigning to mutation

for (i in 2:x) {

df\_new1[,i] <- ifelse(msa\_sel[,1]== msa\_sel[,i],"-", paste0(msa\_sel[,1],">",msa\_sel[,i]))

}

#Load library

library(dplyr) #required for modification of data

#Remove reference genome column and last column (NA column) from the dataframe

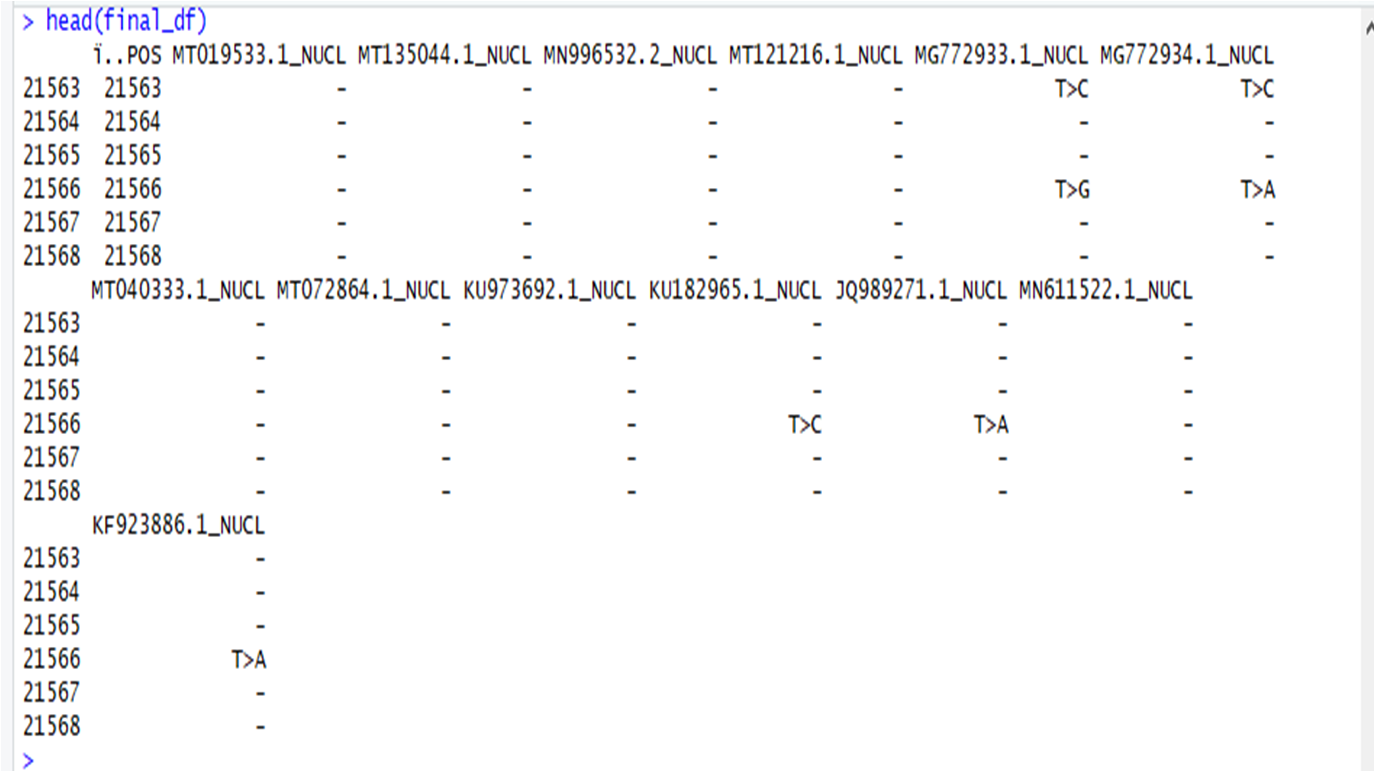
df\_new2 <- select(df\_new1,-1, -ncol(df\_new1))

#Add position column

final\_df <- cbind(pos, df\_new2)

#Check whether output is correct

head(final\_df)



#Write into a file

write.table(final\_df, file="variant\_info.txt", row.names = F, sep ="\t")

## 

## 7. ASSIGNMENT:

Prepare a multiple sequence alignment of several samples and generate a report for ORF6 protein of SARS-COV2