Assignment_6

Abbas Zal

```
library(ggplot2)
library(lubridate)

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(coda)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(rjags)

## Linked to JAGS 4.3.1

## Loaded modules: basemod,bugs
```

execise 1

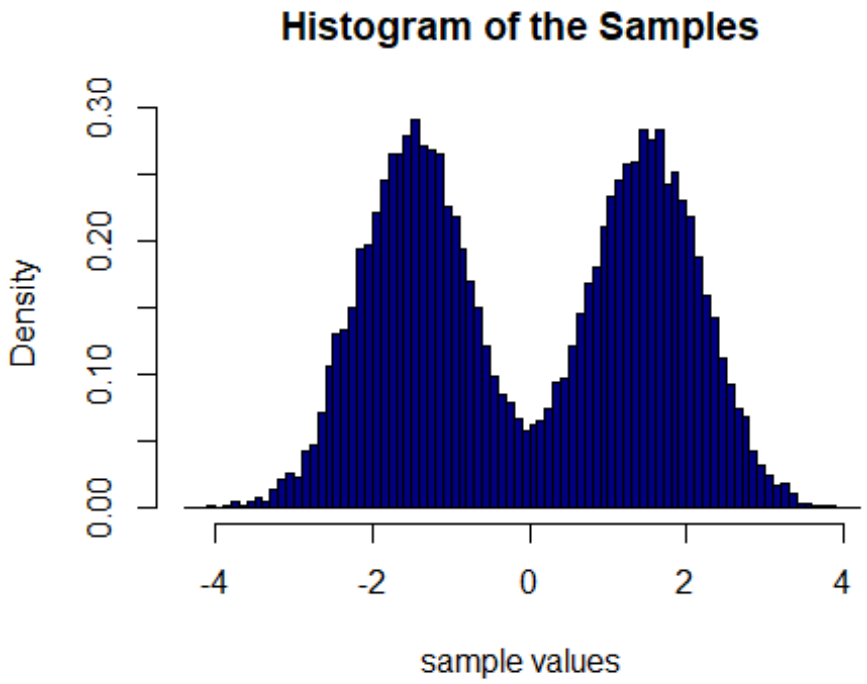
we have the following un-normalized posterior distribution:

```
g <- function(theta) {
  numerator <- exp(-(theta + 3)^2 / 2) + exp(-(theta - 3)^2 / 2)
  denominator <- 2
  result <- numerator / denominator
  return(result)
}
```

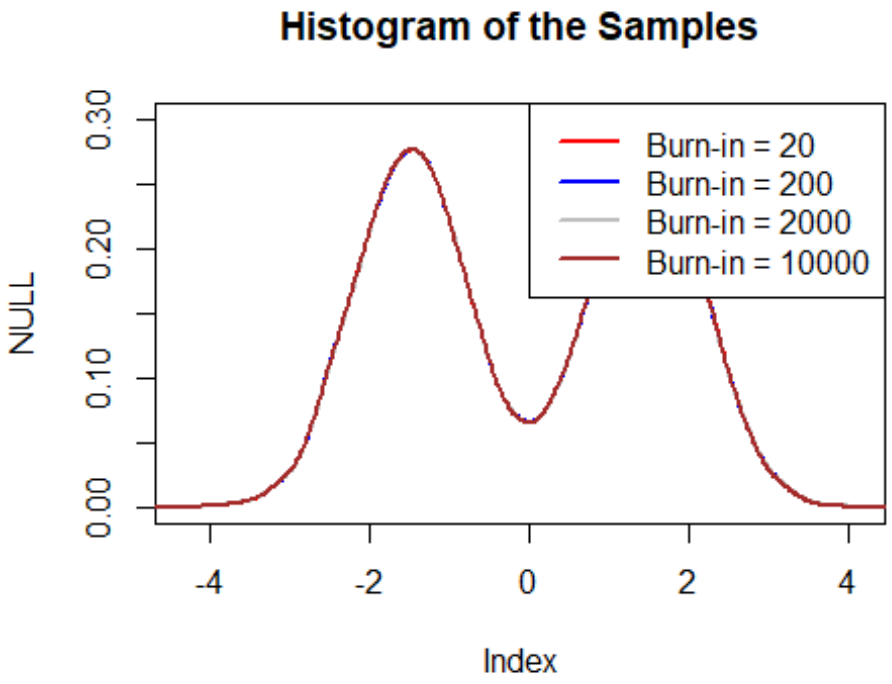
First, we need to document the Metropolis-Hasting algorithm:

```
# Parameters:
# func : a function whose first argument is a real vector of parameters
# func returns a log10 of the likelihood function
```

```r
# theta.init : the initial value of the Markov Chain (and of func)
# n.sample: number of required samples
# sigma : standar deviation of the gaussian MCMC sampling pdf

metropolis.Hasting <- function(func , theta.init , n.sample) {
theta.cur <- theta.init
func.Cur <- func(theta.cur)
func.Samp <- matrix(data=NA, nrow=n.sample , ncol=2+1)
n.accept <- 0
rate.accept <- 0.0

for (n in 1:n.sample) {
theta.prop <- rnorm(n=1, mean = 0, 1)
func.Prop <- func(theta.prop)
logMR <- func.Prop - func.Cur # Log10 of the Metropolis ratio
if ( logMR >=0 || logMR >log10(runif(1)) ) {
theta.cur <- theta.prop
func.Cur <- func.Prop
n.accept <- n.accept + 1
}
func.Samp[n, 1] <- func.Cur
func.Samp[n, 2] <- theta.cur
func.Samp[n, 3] <- n
}
return(func.Samp)
}

g.metropolis <- function(lambda) {
  return(log10(g(lambda)))}


theta.init <- 0.1
n.sample <- 100000

set.seed(20190513)
chain <- metropolis.Hasting(func=g.metropolis ,
                            theta.init = theta.init ,
                            n.sample = n.sample )


n <- nrow(chain)


mcmc.data = chain[1000:n , 2]

hist(mcmc.data , breaks = 100 , freq = F , main = 'Histogram of the Samples'
, xlab = 'sample values', col = "navyblue")
```

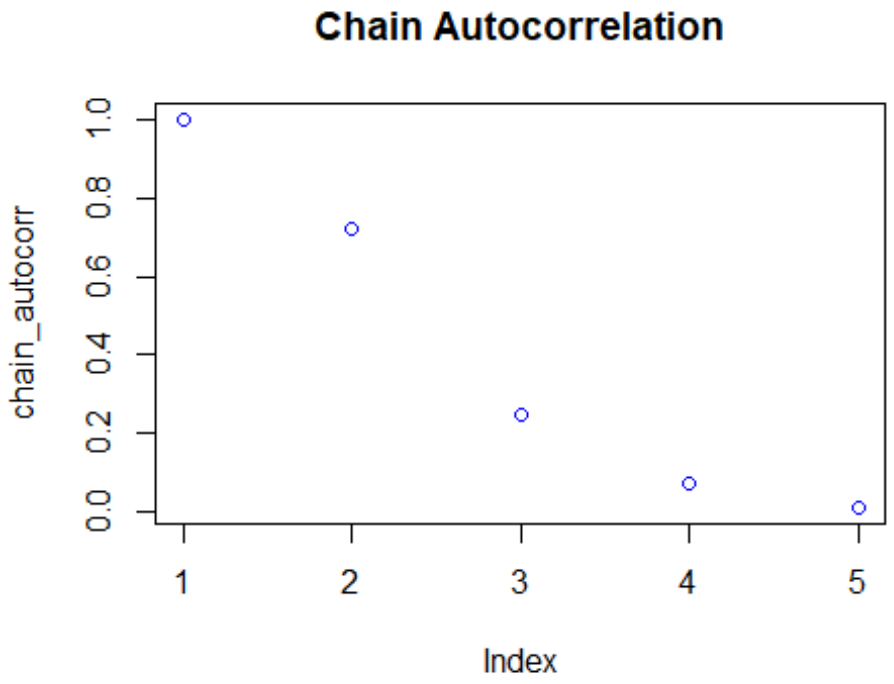



```r
burn_ins <- c(20, 200, 2000, 10000)
legend_labels <- paste("Burn-in =", burn_ins)
colors <- c("red", "blue", "gray", "brown")
# Create an empty plot
plot(NULL, xlim = range(chain[, 2]), ylim = c(0, 0.3), type = "n", main =
'Histogram of the Samples')
     for (i in 1:length(burn_ins)){
       bi <- burn_ins[i]
       mcmc.data <- chain[(bi+1):nrow(chain), 2] # Exclude burn-in samples
       # Compute density estimate
       dens <- density(mcmc.data)
       # Plot density as a line
       lines(dens$x, dens$y, col = colors[i], lwd = 2)
     }
     # Add legend
     legend("topright", legend = legend_labels, col = colors, lwd = 2)
```



Now, we want to analyze the chain with the CODA package and plot the chain autocorrelation:

```
# Convert your chain data to a CODA object
mcmc.data <- as.mcmc(mcmc.data)
# Compute the autocorrelation
chain_autocorr <- autocorr(mcmc.data)
# Plot the chain autocorrelation
plot(chain_autocorr, main = "Chain Autocorrelation" , col = 'blue')
```



In the previous code:The as.mcmc() function is used to convert your chain data to a CODA object. This step is necessary to utilize the functions provided by the CODA package for analyzing and plotting MCMC chains.

The autocorr() function is applied to the converted CODA object mcmc.data to compute the autocorrelation of the chain. Autocorrelation measures the dependence between values in the chain at different lags. It provides insights into the level of correlation and can help determine the appropriate thinning parameter.

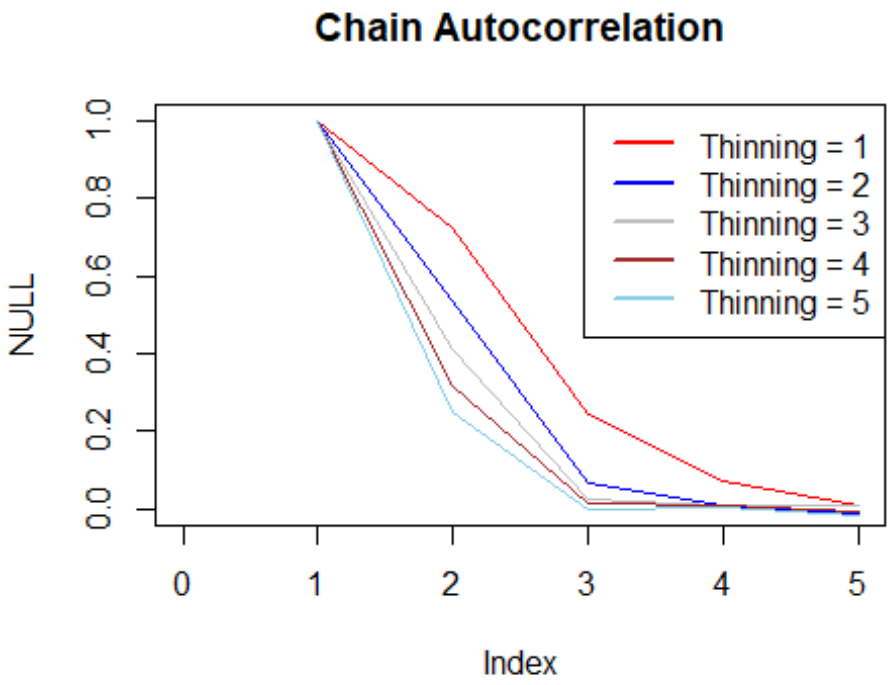
Finally, the plot() function is used to plot the chain autocorrelation. The resulting plot shows the autocorrelation at different lags, with the x-axis representing the lag and the y-axis indicating the autocorrelation value. The main argument is used to specify the main title of the plot, and col is used to set the color of the plot.

By analyzing the chain autocorrelation plot, you can assess the rate at which the correlation decreases with increasing lag. This information can guide your choice of the thinning parameter and help identify the appropriate lag to reduce autocorrelation effectively.

Now,we try to use different burn-in cycles and thinning and plot the corresponding posterior distribution and the chain autocorrelation function.

```
    dif_burn_ins <- c(1, 2, 3, 4, 5)
    colors <- c("red", "blue", "gray", "brown","skyblue")
    legend_labels <- paste("Thinning =", dif_burn_ins)
    # Create an empty plot
    plot(NULL, xlim = c(0, length(chain_autocorr)), ylim = c(0, 1), type =
"n", main = "Chain Autocorrelation")
```

```
    for (i in 1:length(dif_burn_ins)){
      thin <- dif_burn_ins[i]
      mcmc.data <- chain[seq(from = 1000, to = nrow(chain), by = thin), 2]
      mcmc.data <- as.mcmc(mcmc.data)
      chain_autocorr <- autocorr(mcmc.data)
      # Plot autocorrelation with different colors
      lines(chain_autocorr, col = colors[i])
    }
    # Add legend
    legend("topright", legend = legend_labels, col = colors, lwd = 2)
```



##exercise 2

the European Medicines Agency (EMA) has authorized a list of COVID-19 vaccines, after having performed a scientific evaluation of the veccines efficacy, now we want to analyze the initial test data reported on the EMA Web site for the following early Vaccines:

- Moderna - AstraZeneca - Jcovden

and create a Markow Chain Monte Carlo JAGS or stan the efficacy of each Vaccine. Infere the 95% credibility interval.

First of all, Moderna:

```
# Prepare the data
vaccine_cases <- 11
vaccine_population <- 14134
placebo_cases <- 185
```

```r
placebo_population <- 14073
data <- list(vaccine_cases = vaccine_cases,
vaccine_population = vaccine_population,
placebo_cases = placebo_cases,
placebo_population = placebo_population)
# Compile the model
model <- jags.model("model_ex2.txt", data = data)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2
##    Unobserved stochastic nodes: 2
##    Total graph size: 11
##
## Initializing model

#model_ex2 {
  # Priors
  #p_vaccine ~ dbeta(1, 1)
  #p_placebo ~ dbeta(1, 1)

  # Likelihood
  #vaccine_cases ~ dbin(p_vaccine, vaccine_population)
  #placebo_cases ~ dbin(p_placebo, placebo_population)

  # Derived quantities
  #efficacy <- (p_placebo - p_vaccine) / p_placebo * 100
#}

# Define parameters to monitor
parameters <- c("p_vaccine", "p_placebo", "efficacy")
# Run the MCMC
iterations <- 10000
burn_in <- 20000
thin <- 1
chains <- 2
samples <- coda.samples(model, variable.names = parameters,
n.iter = iterations, thin = thin, n.chains = chains)

# Summarize the results
summary(samples)

##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
```

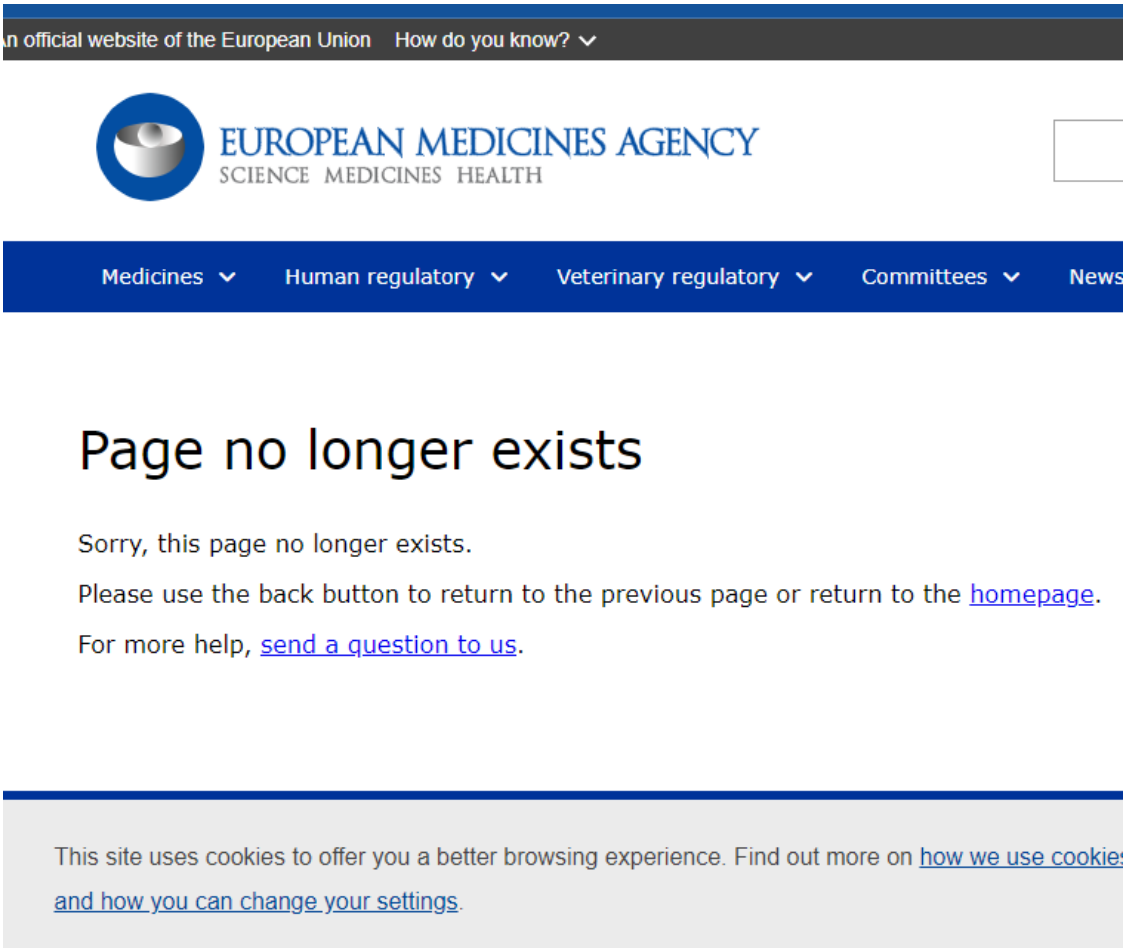
```
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## efficacy  9.356e+01 1.9277921 1.928e-02      2.684e-02
## p_placebo 1.322e-02 0.0009640 9.640e-06      1.263e-05
## p_vaccine 8.463e-04 0.0002434 2.434e-06      3.418e-06
##
## 2. Quantiles for each variable:
##
##                2.5%       25%       50%       75%     97.5%
## efficacy  8.925e+01 9.240e+01 9.376e+01 94.969997 96.696630
## p_placebo 1.142e-02 1.256e-02 1.320e-02  0.013860  0.015138
## p_vaccine 4.392e-04 6.706e-04 8.211e-04  0.000999  0.001383
```

Second of all, Jcovden: as same as Moderna. #Jcovden:

```
#as i saw in link:
# Prepare the data
vaccine_cases <- 116
vaccine_population <- 19630
placebo_cases <- 348

placebo_population <- 19691
data <- list(vaccine_cases = vaccine_cases,
vaccine_population = vaccine_population,
placebo_cases = placebo_cases,
placebo_population = placebo_population)
# Compile the model
model <- jags.model("model_ex2.txt", data = data)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2
##    Unobserved stochastic nodes: 2
##    Total graph size: 11
##
## Initializing model

# Define parameters to monitor
parameters <- c("p_vaccine", "p_placebo", "efficacy")
# Run the MCMC
iterations <- 10000
burn_in <- 20000
thin <- 1
chains <- 2
samples <- coda.samples(model, variable.names = parameters,
n.iter = iterations, thin = thin, n.chains = chains)
```

```
# Summarize the results
summary(samples)

##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## efficacy  66.298417 3.5956052 3.596e-02      4.392e-02
## p_placebo  0.017718 0.0009397 9.397e-06      1.223e-05
## p_vaccine  0.005955 0.0005522 5.522e-06      6.597e-06
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## efficacy  58.98155 63.941078 66.479953 68.831426 72.824614
## p_placebo  0.01594  0.017084  0.017700  0.018332  0.019620
## p_vaccine  0.00493  0.005569  0.005935  0.006322  0.007086
```

AstraZeneca's link was broken.



exercise 3

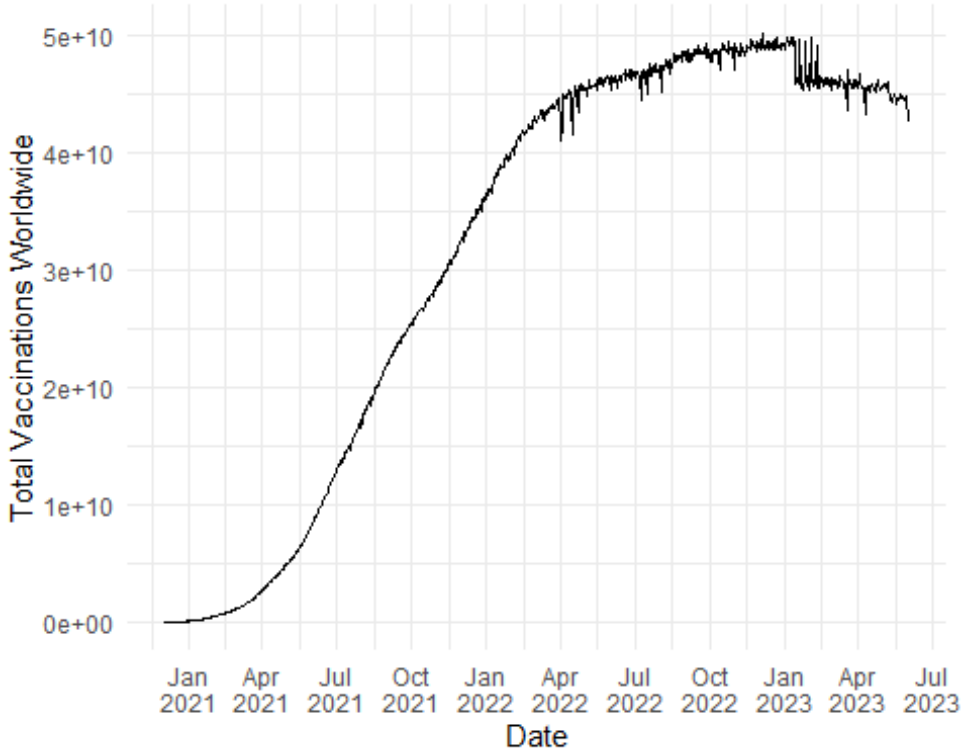
According to the official COVID-19 vaccination data, 70% of the world population has received at least one dose of a COVID-19 vaccine.

Now we want to analyze the data and produce the following plots:

- number of vaccinated people (cumulative, daily and week average)
- number of confirmed deaths by COVID-19, both cumulative and weekly average

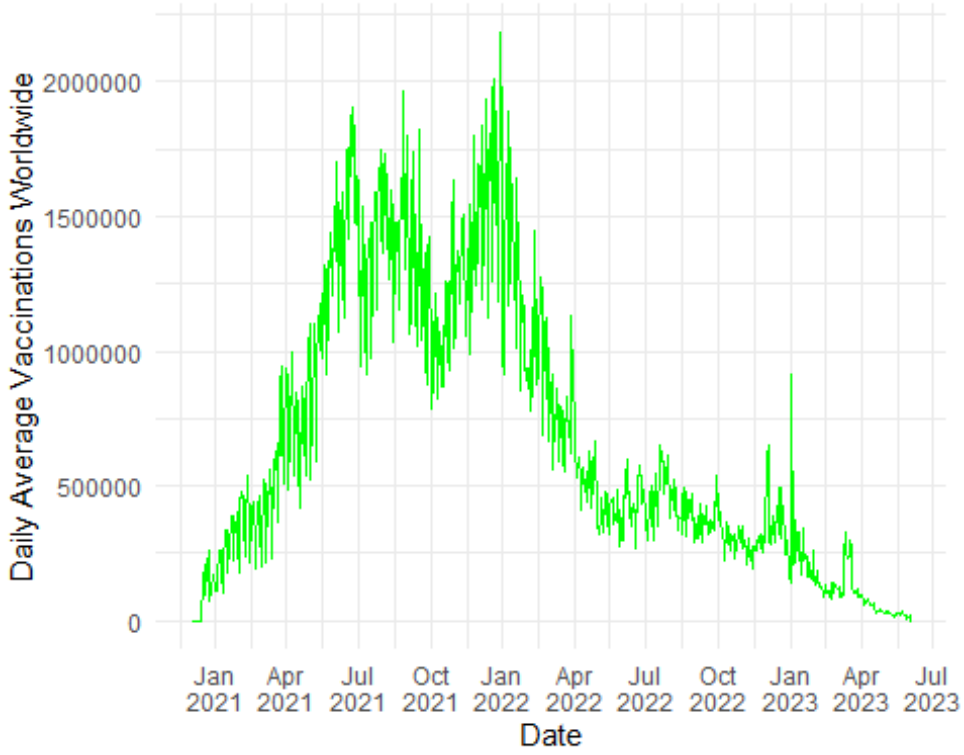
```
#Loading the data
# Set the working directory to the folder containing the CSV file
setwd("F:/R/final6")

# Read the CSV file
global_data <- read.csv("covid-data.csv")
#str(global_data)

# Filter the data to include only the relevant columns
filtered_data <- global_data[, c("date", "total_vaccinations")]
```

```r
# Filter out rows with missing or zero total_vaccinations
filtered_data <- filtered_data[!is.na(filtered_data$total_vaccinations) &
filtered_data$total_vaccinations > 0, ]

# Calculate the cumulative vaccinations worldwide
worldwide_data <- filtered_data %>%
  group_by(date) %>%
  summarise(total_vaccinations_worldwide = sum(total_vaccinations))

# Convert date column to date format
worldwide_data$date <- as.Date(worldwide_data$date)

# Plot the total number of vaccinated people worldwide over time
plot_cumulative <- ggplot(worldwide_data, aes(x = date, y =
total_vaccinations_worldwide)) +
  geom_line() +
  labs(x = "Date", y = "Total Vaccinations Worldwide") +
  scale_x_date(date_labels = "%b\n%Y", date_breaks = "3 month") +
  theme_minimal()

# Display the cumulative plot
print(plot_cumulative)
```



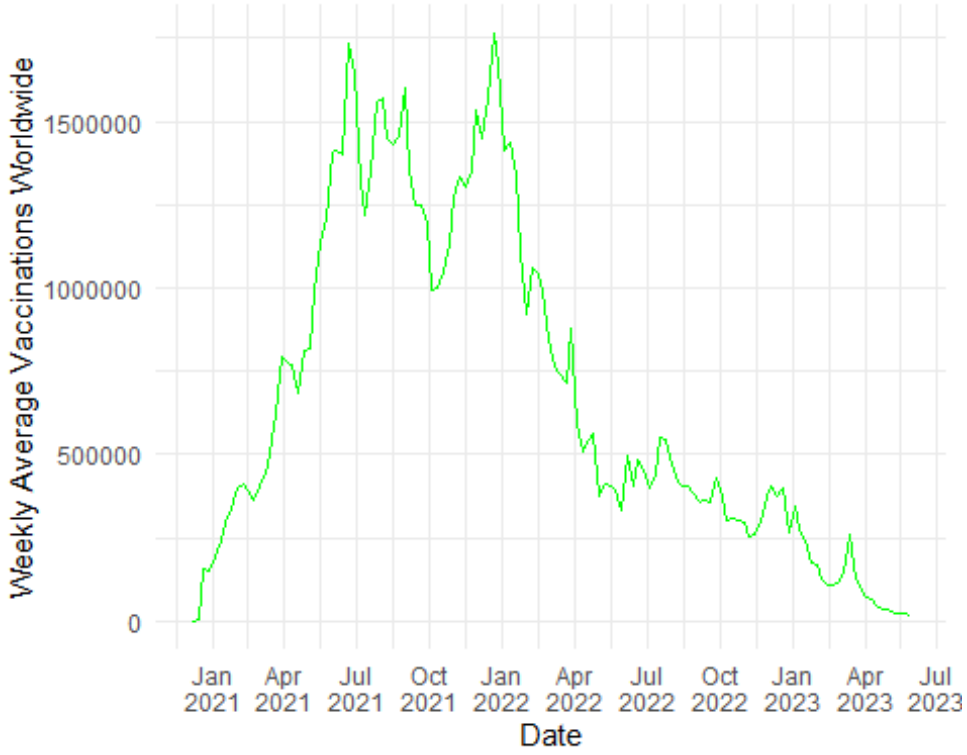

```r
# Filter the global_data to include only the relevant columns and remove
missing or zero new_vaccinations
worldwide_data <- global_data %>%
```

```
  select(date, new_vaccinations) %>%
  filter(!is.na(new_vaccinations), new_vaccinations != 0) %>%
  mutate(date = as.Date(date, format = "%Y-%m-%d")) %>%
  group_by(date) %>%
  summarise(average_vaccinations_worldwide = mean(new_vaccinations))

# Plot the daily average vaccinations worldwide over time
plot <- ggplot(worldwide_data, aes(x = date, y =
average_vaccinations_worldwide)) +
  geom_line(color='green') +
  labs(x = "Date", y = "Daily Average Vaccinations Worldwide") +
  scale_x_date(date_labels = "%b\n%Y", date_breaks = "3 month") +
  theme_minimal()

# Display the plot
print(plot)
```

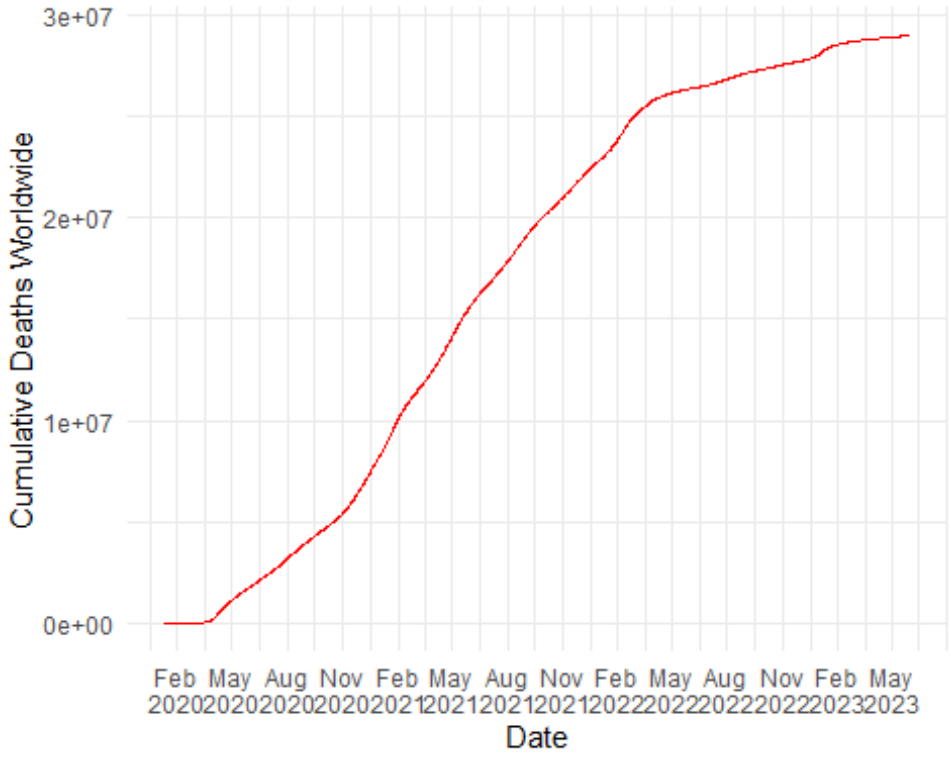



```
filtered_data <- global_data[, c("date", "new_vaccinations") , drop = FALSE]

filtered_data$date <- as.Date(filtered_data$date, format = "%Y-%m-%d")
filtered_data <- filtered_data[!is.na(filtered_data$new_vaccinations) &
filtered_data$new_vaccinations != 0, ]

# Extract the year and week from the date
filtered_data$year <- year(filtered_data$date)
filtered_data$week <- week(filtered_data$date)
```

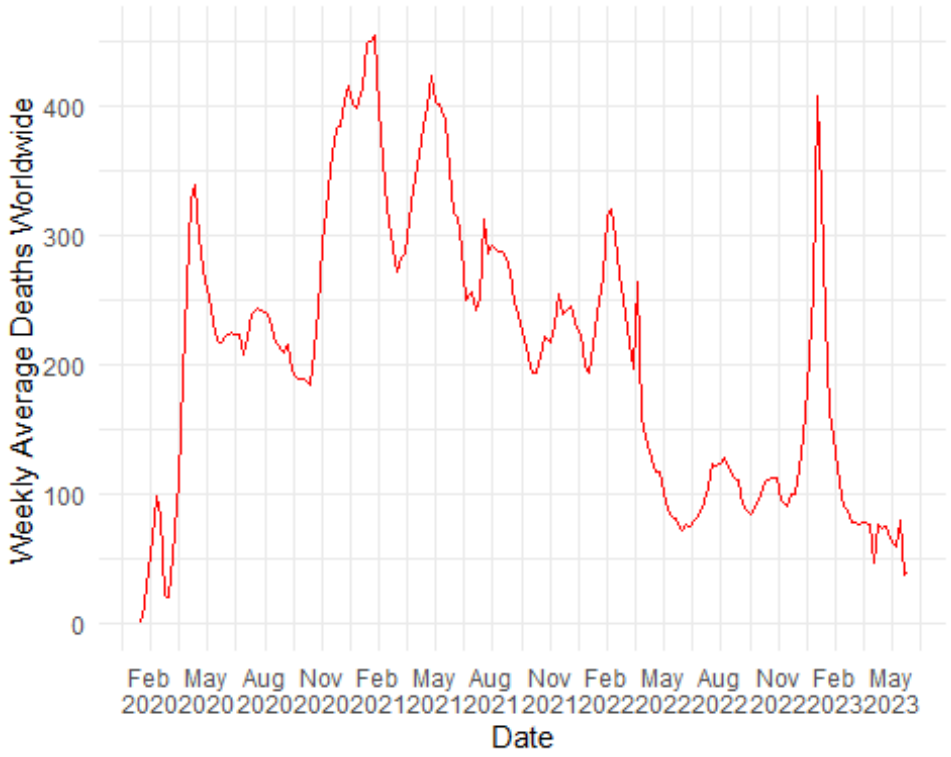
```r
# Calculate the weekly average vaccinations worldwide
worldwide_data <- filtered_data %>%
  group_by(year, week) %>%
  summarise(average_vaccinations_worldwide = mean(new_vaccinations))

## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.

# Create a new date column representing the start of each week
worldwide_data$date <- as.Date(paste(worldwide_data$year,
worldwide_data$week, "1", sep = "-"), format = "%Y-%U-%u")

## Warning in strptime(x, format, tz = "GMT"): (0-based) yday 369 in year
2020 is
## invalid

## Warning in strptime(x, format, tz = "GMT"): (0-based) yday 367 in year
2021 is
## invalid

## Warning in strptime(x, format, tz = "GMT"): (0-based) yday 366 in year
2022 is
## invalid

# Plot the weekly average vaccinations worldwide over time (with green color)
plot <- ggplot(worldwide_data, aes(x = date, y =
average_vaccinations_worldwide)) +
  geom_line(color = "green") +
  labs(x = "Date", y = "Weekly Average Vaccinations Worldwide") +
  scale_x_date(date_labels = "%b\n%Y", date_breaks = "3 months") +
  theme_minimal()

# Display the plot
print(plot)

## Warning: Removed 3 rows containing missing values (`geom_line()`).
```




```r
# Filter the data to include only the relevant columns and remove missing or
zero total_deaths
filtered_data <- global_data[,c("date", "total_deaths") , drop = FALSE]
filtered_data$date <- as.Date(filtered_data$date, format = "%Y-%m-%d")
filtered_data <- filtered_data[!is.na(filtered_data$total_deaths) &
filtered_data$total_deaths != 0, ]

# Calculate the cumulative number of deaths worldwide
worldwide_data <- filtered_data %>%
  group_by(date) %>%
  summarise(cumulative_deaths_worldwide = sum(total_deaths))

# Plot the cumulative number of deaths worldwide over time (with red color)
plot <- ggplot(worldwide_data, aes(x = date, y =
cumulative_deaths_worldwide)) +
  geom_line(color = "red") +
  labs(x = "Date", y = "Cumulative Deaths Worldwide") +
  scale_x_date(date_labels = "%b\n%Y", date_breaks = "3 months") +
  theme_minimal()

# Display the plot
print(plot)
```




```r
# Filter the data to include only the relevant columns and remove missing or
zero new_deaths
filtered_data <- global_data[,c("date", "new_deaths") , drop = FALSE]
filtered_data$date <- as.Date(filtered_data$date, format = "%Y-%m-%d")
filtered_data <- filtered_data[!is.na(filtered_data$new_deaths) &
filtered_data$new_deaths != 0, ]

# Calculate the weekly average number of deaths worldwide
worldwide_data <- filtered_data %>%
  group_by(date = as.Date(floor_date(date, "week"))) %>%
  summarise(average_deaths_worldwide = mean(new_deaths))

# Plot the weekly average number of deaths worldwide over time (with red
color)
plot <- ggplot(worldwide_data, aes(x = date, y = average_deaths_worldwide)) +
  geom_line(color = "red") +
  labs(x = "Date", y = "Weekly Average Deaths Worldwide") +
  scale_x_date(date_labels = "%b\n%Y", date_breaks = "3 months") +
  theme_minimal()

# Display the plot
print(plot)
```


Weekly Average Deaths Worldwide
400
300
200
100
0
Feb 2020
May 2020
Aug 2020
Nov 2020
Feb 2021
May 2021
Aug 2021
Nov 2021
Feb 2022
May 2022
Aug 2022
Nov 2022
Feb 2023
May 2023
Date