Exercise 6 - Advanced Statistics for Physics Analysis (A.Y. 2022-2023)

by Erica Brisigotti (2097202)

Exercise 1

Given the following un-normalized posterior distribution

$$g(\theta|x) \propto \frac{1}{2}\exp\left(-\frac{(\theta+3)^2}{2}\right) + \frac{1}{2}\exp\left(-\frac{(\theta-3)^2}{2}\right)$$

I find the norm of the un-normalized posterior by integrating analytically

$$|g| = \int_{-\infty}^{+\infty} \exp(\left(-\frac{(\theta+3)^2}{2}\right))\, d\theta + \int_{-\infty}^{+\infty} \exp(\left(-\frac{(\theta'-3)^2}{2}\right))\, d\theta'$$

By applying two changes of variable $\theta + 3 \to t$ and $\theta' + 3 \to s$, I notice that the two integrals correspond to the same Gaussian integral:

$$|g| = \int_{-\infty}^{+\infty} \exp(\left(-\frac{t^2}{2}\right))\, dt + \int_{-\infty}^{+\infty} \exp(\left(-\frac{s^2}{2}\right))\, ds = 2\int_{-\infty}^{+\infty} \exp(\left(-\frac{s^2}{2}\right))\, ds = 2\sqrt{2\pi}$$

In [1]:
```
# I define the normalized posterior distribution
# by initializing the norm of the un-normalized posterior distribution
log_post_int <- 2*(2*pi)**0.5
# and dividing the un-normalized posterior distribution by said norm
log_post <- function(theta){ log(0.5 * exp(-(theta + 3)^2 / 2) + 0.5 * exp(-(theta - 3)^2 / 2)) /log_post_int }
```

- draw a Markov Chain from the posterior distribution using a Metropolis-Hastings algorithm

In [2]:
```
# I assume a uniform distribution of the proposed values
proposal <- function(theta, sigma){ (rnorm(1, mean = theta, sd = sigma)) }
```

In [3]:
```
# and initialize the function that executes the Metropolis-Hastings algorithm
metropolis_hastings <- function(num_iterations, initial_theta, scale) {
    # I set its starting values
    theta <- initial_theta
    # and initialize the variables that will be modified during the iteration, which are
    # an array in which I'm going to save the values of theta
    chain <- numeric(num_iterations)
    # and a counter for the number of accepted values of theta
    accepted <- 0
    # At every iteration
    for (i in 1:num_iterations) {
        # I propose a value for the parameter theta based on the previously defined proposal function
        proposed_theta <- proposal(theta, scale)
        # I calculate the logarithm of the ration betweent the posterior probabilities
        log_ratio <- log_post(proposed_theta) - log_post(theta)
        # and ensure that the probability is less or equal to 1
        acceptance_prob <- min(1, exp(log_ratio))
        # I then extract a random number
        u <- runif(1)
        # based on which I decide weather or not to accept the value of theta or not
        if (u < acceptance_prob){
            # whenever I accept, I update the value of theta to the one proposed
            theta <- proposed_theta
            # and update the updated counter
            accepted <- accepted + 1 }
        # otherwise (if I don't accept) the values of theta and counter are unchanged
        chain[i] <- theta }
    # lastly, I calculate the acceptance probability
    acceptance_rate <- accepted / num_iterations
    # and structure the data that I want to store in a list (because of lists flexibility)
    result <- list(chain = chain, acceptance_rate = acceptance_rate)
    return(result) }
```

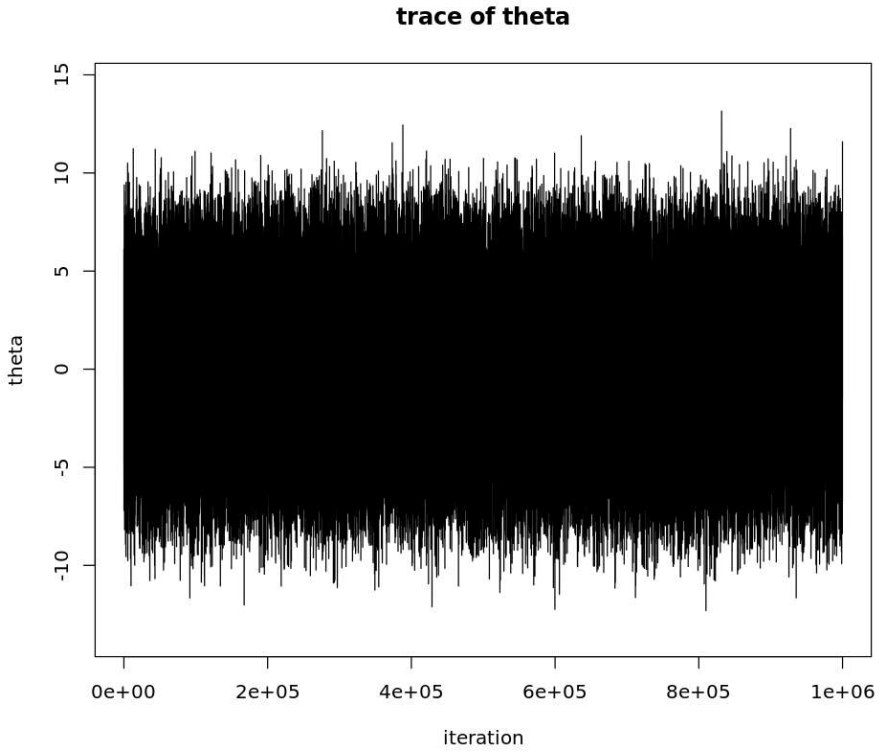
- use a Norm$(0, 1)$ as random-walk candidate density

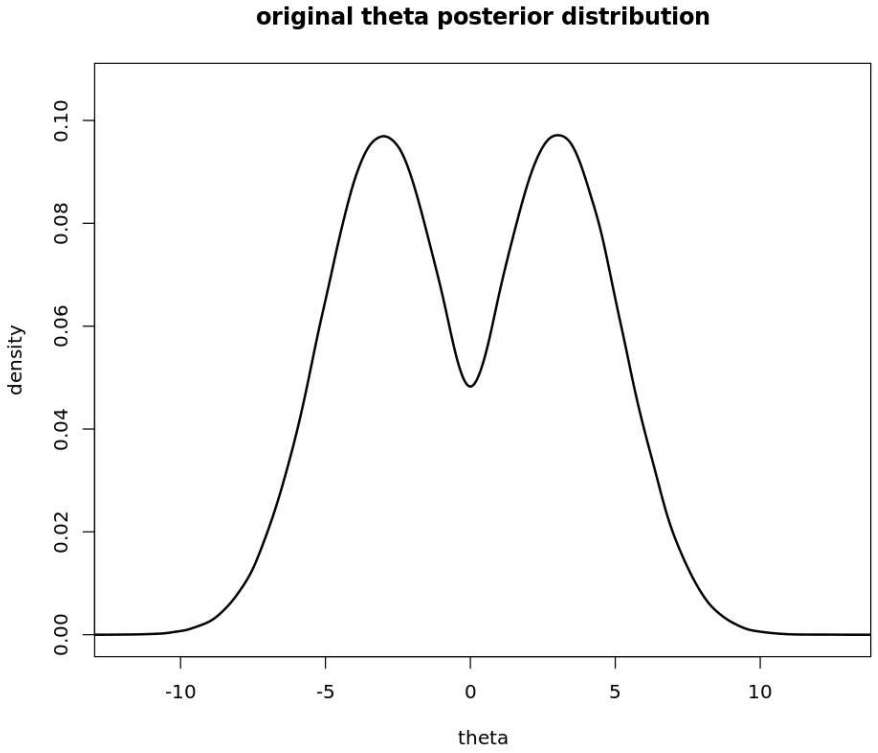
In [4]:
```
# now, I just have to set the parameters of my simulation
num_iterations <- 1000000
initial_theta <- 0
sigma <- 1
# and finally run it
result <- metropolis_hastings(num_iterations, initial_theta, sigma)
# the results are the following
cat("the acceptance rate of the MCMC algorithm is", result$acceptance_rate*100, "%. \n")
```

```
the acceptance rate of the MCMC algorithm is 88.665 %.
```

- plot the sampled distribution

In [5]:
```
# I decide to represent the trace of th MCMC
options(repr.plot.width=16, repr.plot.height=7)
par( mfrow=c(1,2) )
plot(result$chain, type = "l", xlab = "iteration", ylab = "theta", col="black", ylim=1.1*c(min(result$chain),max(result$chain)
     main = "trace of theta")
# and the density/distribution of the MCMC parameter theta
dens <- density(result$chain)
plot(dens$x, dens$y, type="l", lwd=2, xaxs="i", col="black", ylim=1.1*c(0,max(dens$y)), xlab='theta', ylab="density",
    main = "original theta posterior distribution")
```



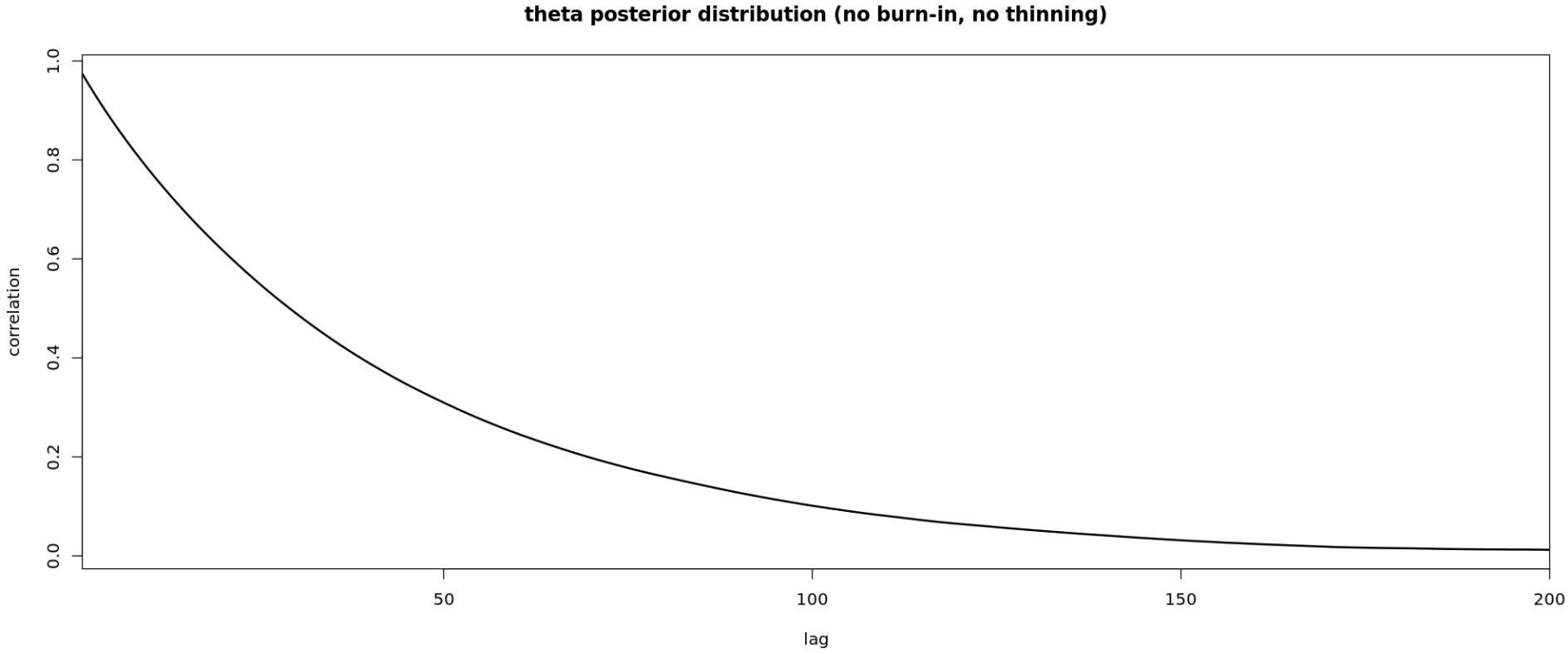


- analyze the chain with the CODA package and plot the chain autocorrelation

In [6]:
```
library(coda)
```

In [7]:
```
# I save my chain in a dataframe format
c <- as.mcmc( result$chain )
# and get a summary statistics of the parameter of the MCMC chain
summary(c)
# and plot the chain autocorrelation plot
options(repr.plot.width=16, repr.plot.height=7)
ac <- autocorr(c, lags = 1:200)
plot(ac, type="l", lwd=2, xaxs="i", col="black", xlab='lag', ylab="correlation",
    main = "theta posterior distribution (no burn-in, no thinning)")
```

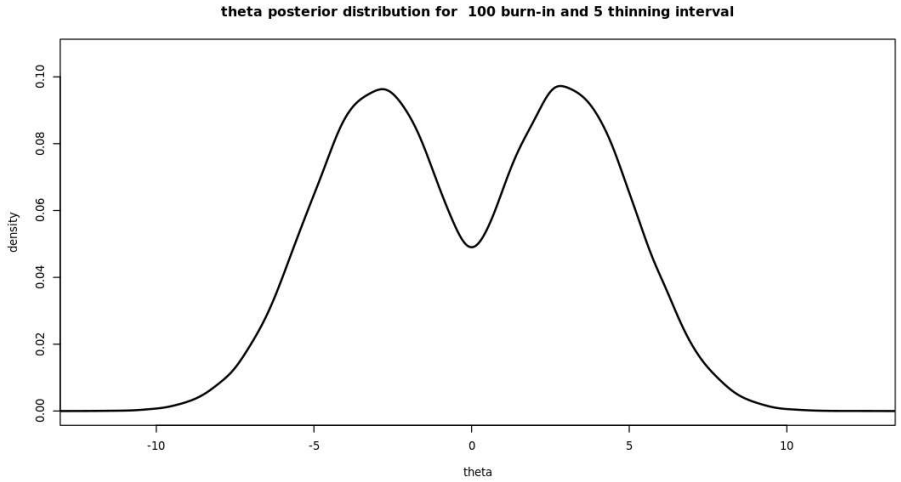
```
Iterations = 1:1e+06
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1e+06

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
    -0.0001732      3.8758881      0.0038759      0.0356186

2. Quantiles for each variable:

      2.5%        25%        50%        75%      97.5%
-6.7633337 -3.2319083  0.0001085  3.2367856  6.7333728
```

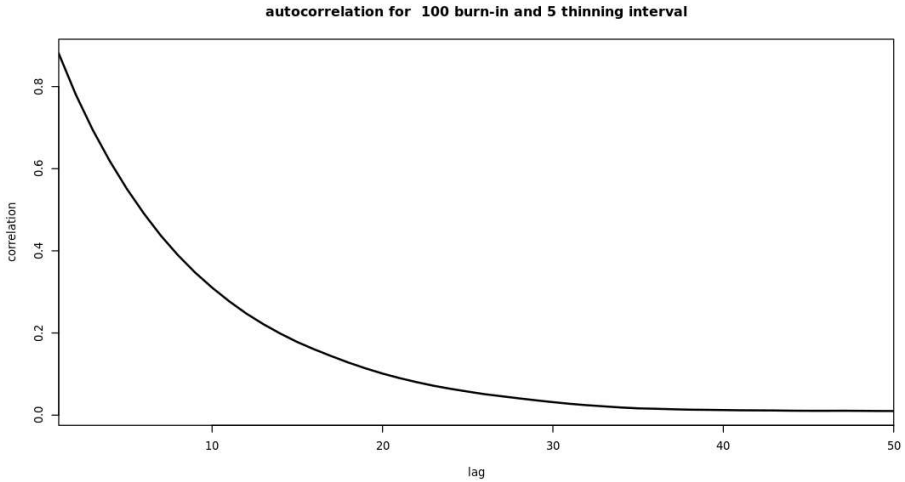


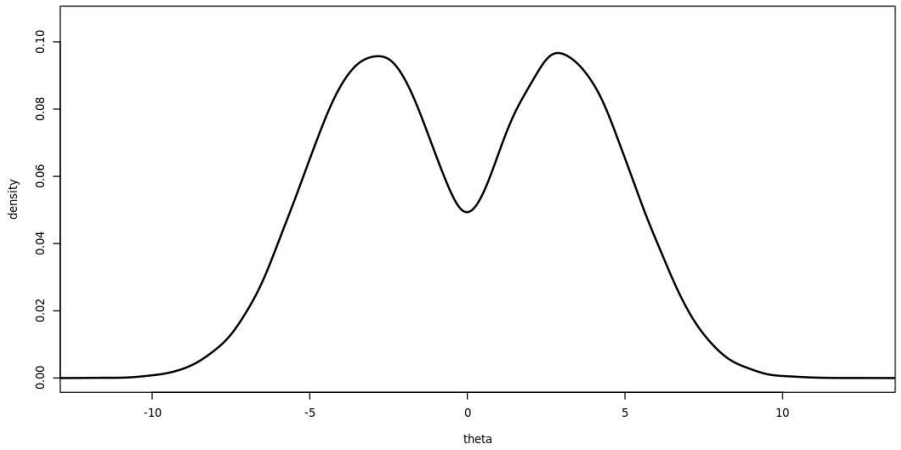
The original chain could be less auto-correlated.

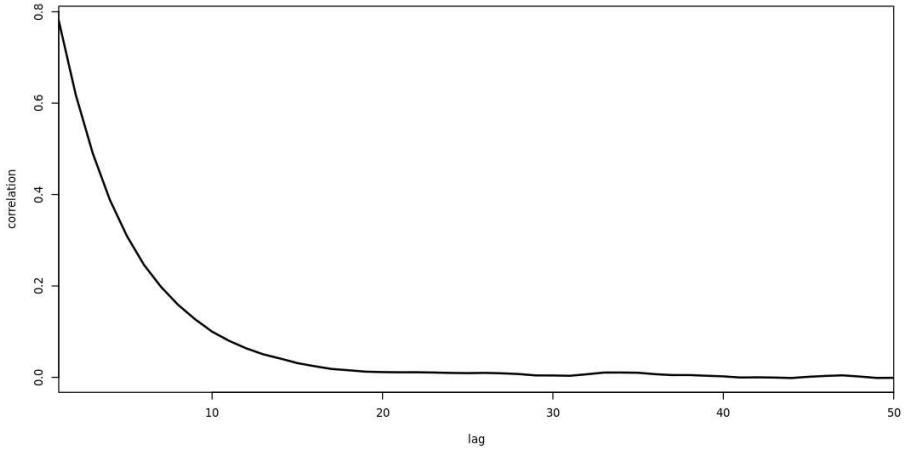
- try to use different burn-in cycles and thinning and plot the corresponding posterior distribution and the chain autocorrelation function. What are the best parameters ?

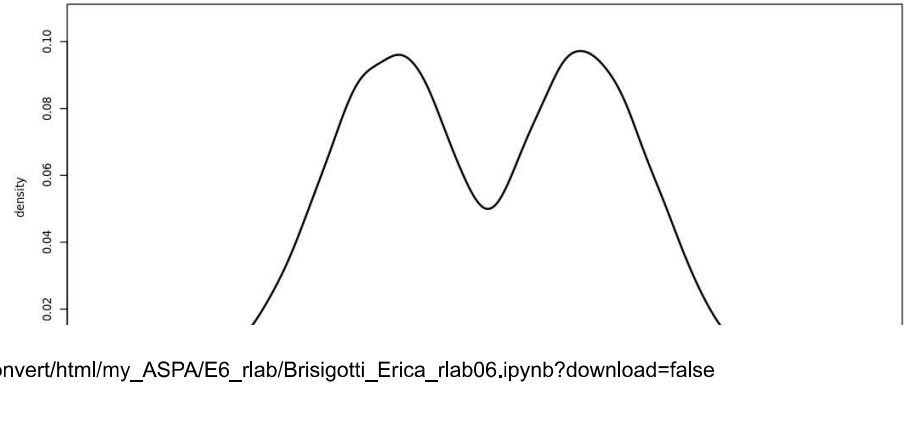
In [8]:
```
# I select some values for the burn-in
burn_in_values = c(0.01, 0.1, 1)*num_iterations/100
# and for the thinning intervals
thinning_intervals = c(0.0005, 0.001, 0.0025)*num_iterations/100

# I configure the plots
options(repr.plot.width=18, repr.plot.height=15)
par( mfrow=c(3,2) )
# for every combination of parameters
for (i in 1:3){
    # I crop the burn-in
    temp <- result$chain[seq(burn_in_values[i]+1,length(result$chain),1)]
    # and thin the burned vector
    idx <- seq(1, length(temp), thinning_intervals[i])
    temp <- temp[idx]
    # I plot the posterior distribution
    dens <- density(temp)
    plot(dens$x, dens$y, type="l", lwd=2, xaxs="i", col="black", ylim=1.1*c(0,max(dens$y)), xlab='theta', ylab="density",
        main = paste("theta posterior distribution for ",burn_in_values[i],'burn-in and',thinning_intervals[i],
                     'thinning interval'))
    # and the auto-correlation function
    c <- as.mcmc( temp )
    ac <- autocorr(c, lags = 1:50)
    plot(ac, type="l", lwd=2, xaxs="i", col="black", xlab='lag', ylab="correlation",
        main = paste("autocorrelation for ",burn_in_values[i],'burn-in and',thinning_intervals[i],
                     'thinning interval'))
}
```

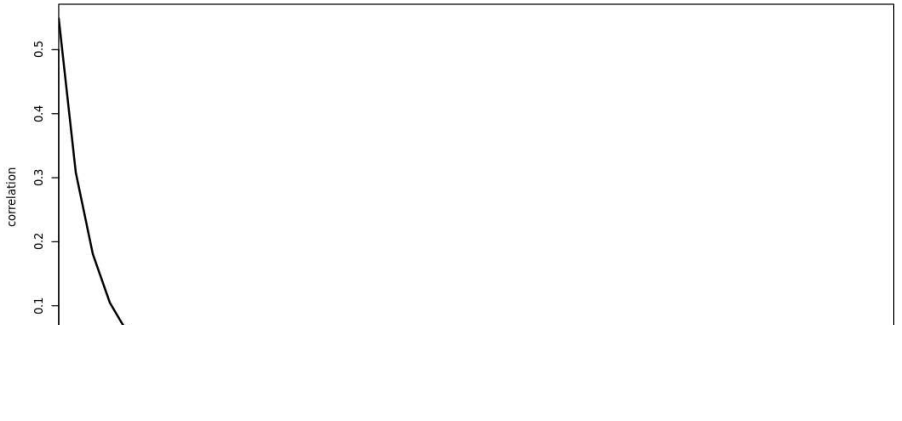

theta posterior distribution for 100 burn-in and 5 thinning interval
density
theta

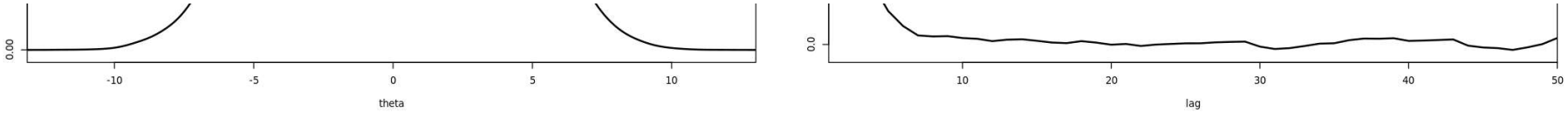

autocorrelation for 100 burn-in and 5 thinning interval
correlation
lag


theta posterior distribution for 1000 burn-in and 10 thinning interval
density
theta


autocorrelation for 1000 burn-in and 10 thinning interval
correlation
lag


theta posterior distribution for 10000 burn-in and 25 thinning interval
density


autocorrelation for 10000 burn-in and 25 thinning interval
correlation



The burn-in and thinning strategy definitely helped reduce the auto-correlatation.

Exercise 2

The European Medicines Agency (EMA) has authorized a list of COVID-19 vaccines, after having performed a scientific evaluation of the vaccines efficacy. The following vaccines are currently authorized for use in the European Union:

- Comirnaty (BioNTech and Pfizer)
- VCOVID-19 Vaccine Valneva
- Nuvaxovid (Novavax)
- Pikevax (Moderna)
- Vaxzeviria (AstraZeneca)
- Jcovden (Janssen)
- VidPrevtyn Beta (Sanofi Pasteur)
- Bimervax, previously COVID-19 Vaccine HIPRA (HIPRA Human Health S.L.U.)

Analyze the initial test data reported on the EMA Web site for the following early Vaccines

- Janssen [1]
- Moderna [2]
- AstraZeneca [3]
- Jcovden [4]

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

I notice online that Janssen seems to be just the old name for Jcovden, so I will have to analyze just three cases instead of four.I summarize the analysis into a function to make the code more readable, and apply it to each case later on.

In [9]:
```
library(tidyverse)
library(rjags)
library(bayestestR)
library(logspline)
```

```
── Attaching core tidyverse packages ──────────────────────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.2     ✓ readr     2.1.4
✓ forcats   1.0.0     ✓ stringr   1.5.0
✓ ggplot2   3.4.2     ✓ tibble    3.2.1
✓ lubridate 1.9.2     ✓ tidyr     1.3.0
✓ purrr     1.0.1
── Conflicts ────────────────────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
Linked to JAGS 4.3.0

Loaded modules: basemod,bugs
```

In [10]:
```
# I initialize the model without external files
model_text <- "
    model {
    # it's a bernoulli inference problem, so I use the bernoulli distribution
    # to compute the likelihood
    for ( i in 1:Ntot ) {
    tested[i] ~ dbern( theta[patient[i]] )
    }
    # I do not use a flat prior, since we know how likely is to get COVID.
    # I use a beta(3,100) prior:
    for ( k in 1:Nclass ) {
        theta[k] ~ dbeta(3 , 100)
        }
    }"
```

In [11]:
```
my_Vaccine_analysis <- function(string){
    # for Moderna
    if (string == 'Moderna'){
        tot_vaccine <- 14134
        tot_placebo <- 14073
```

```
    pos_vaccine <- 11
    pos_placebo <- 185
# for AstraZeneca
} else if (string == 'AstraZeneca'){
    tot_vaccine <- 5258
    tot_placebo <- 5210
    pos_vaccine <- 64
    pos_placebo <- 154
# for Jcovden aka Janssen
} else if (any(string == c('Jcovden','Janssen'))) {
    tot_vaccine <- 19630
    tot_placebo <- 19691
    pos_vaccine <- 116
    pos_placebo <- 348
} else {
    stop('no info about this vaccine, please initialize in the function')
}
patient <- c(rep("Vaccine", tot_vaccine),
             rep("Placebo", tot_placebo))
tested <- c(rep("Pos", pos_vaccine),
            rep("Neg", tot_vaccine - pos_vaccine),
            rep("Pos", pos_placebo),
            rep("Neg", tot_placebo - pos_placebo))
tb <- tibble(tested = tested , patient=patient)

# I print the pieces of info about the plot
print(table(tb[[2]], tb[[1]]))

# I initialize the data for the MCMC model
data = list(
    tested = ifelse(tb$tested == "Neg", 0, 1),
    patient = as.integer(factor(tb$patient)),
    Ntot = nrow(tb) ,
    Nclass = nlevels(factor(tb$patient )) )
# initialize such model through rjags
model <- jags.model( file = textConnection(model_text),
                     data = data,
                     quiet = TRUE )
# and run the model to get the samples to analyze
chains <- coda.samples( model,
```

```
                        var = "theta",
                        n.iter = 50000 )
# I transform the output into a dataframe for the custom plots
df <- as.data.frame( as.mcmc(chains) )
colnames(df) <- c('Placebo', 'Vaccine')
# and display a quick summary statistics of the results
temp <- summary(chains)
print(temp)

# I intialize the plot
options(repr.plot.width=18, repr.plot.height=18)
par( mfrow=c(3,2) )
# I plot the trace and average of the Placebo
mean <- temp$statistics['theta[1]','Mean']
plot(df$Placebo, type = "l", xlab = "iteration", ylab = "theta", col="black", ylim=c(min(df$Placebo),max(df$Placebo)),
     xaxs="i", main = paste("trace of theta for Placebo patients -", string, 'vaccine'))
abline(h=mean, col='red', lwd=2, lty=2)
legend('topright', legend=c("trace","average"), col=c('black','red'), lty = c(1, 2), lwd=c(1, 2), cex = 0.8)
# and of the Vaccine
mean <- temp$statistics['theta[2]','Mean']
plot(df$Vaccine, type = "l", xlab = "iteration", ylab = "theta", col="black", ylim=c(min(df$Vaccine),max(df$Vaccine)),
     xaxs="i", main = paste("trace of theta for Vaccine patients -", string, 'vaccine'))
abline(h=mean, col='red', lwd=2, lty=2)
legend('topright', legend=c("trace","average"), col=c('black','red'), lty = c(1, 2), lwd=c(1, 2), cex = 0.8)
# I plot the density and 95% CI limits for the Placebo
dens <- density( df$Placebo )
low_lim <- temp$quantiles['theta[1]','2.5%']
up_lim <- temp$quantiles['theta[1]','97.5%']
plot(dens$x, dens$y, type="l", lwd=2, xaxs="i", col="black", ylim=1.1*c(0,max(dens$y)), xlab='theta', ylab="density",
    main = paste('theta distribution for Placebo patients -', string, 'vaccine'))
abline(v=low_lim, col='green', lwd=2, lty=2)
abline(v=up_lim, col='green', lwd=2, lty=2)
legend('topright', legend=c("trace","95% CI"), col=c('black','green'), lty = c(1, 2), lwd=c(1, 2), cex = 0.8)
# and for the Vaccine
dens <- density( df$Vaccine )
low_lim <- temp$quantiles['theta[2]','2.5%']
up_lim <- temp$quantiles['theta[2]','97.5%']
plot(dens$x, dens$y, type="l", lwd=2, xaxs="i", col="black", ylim=1.1*c(0,max(dens$y)), xlab='theta', ylab="density",
    main = paste('theta distribution for Vaccine patients -', string, 'vaccine'))
abline(v=low_lim, col='green', lwd=2, lty=2)
```

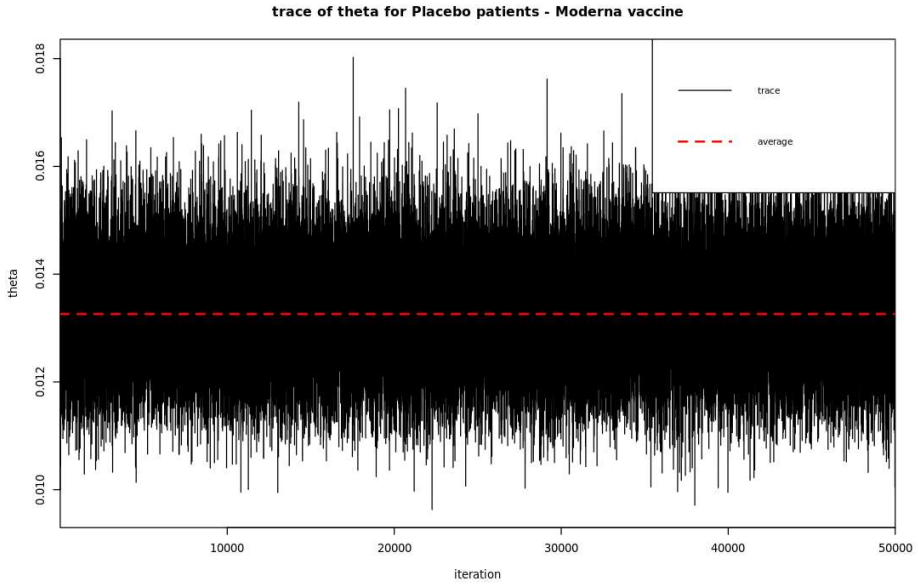
```
    abline(v=up_lim, col='green', lwd=2, lty=2)
    legend('topright', legend=c("trace","95% CI"), col=c('black','green'), lty = c(1, 2), lwd=c(1, 2), cex = 0.8)
    # I plot the autocorrelatin for the Placebo
    c <- as.mcmc( df$Placebo )
    ac <- autocorr(c, lags = 1:50)
    plot(ac, type="l", lwd=2, xaxs="i", col="black", xlab='lag', ylab="correlation",
        main = paste('theta autocorrelation for Placebo patients -', string, 'vaccine'))
    # and for the Vaccine
    c <- as.mcmc( df$Vaccine )
    ac <- autocorr(c, lags = 1:50)
    plot(ac, type="l", lwd=2, xaxs="i", col="black", xlab='lag', ylab="correlation",
        main = paste('theta autocorrelation for Vaccine patients -', string, 'vaccine'))

    # to have quantitative results, I use the bayes factor
    df['diff_rate'] <- (df$Placebo - df$Vaccine) / df$Placebo * 100
    prior <- bayestestR::distribution_normal(50000, mean = 50, sd = 15)
    t <- bayestestR::bayesfactor_parameters(df$diff_rate, prior, direction = "two-sided", null = 50)
    print(t)
    }
```

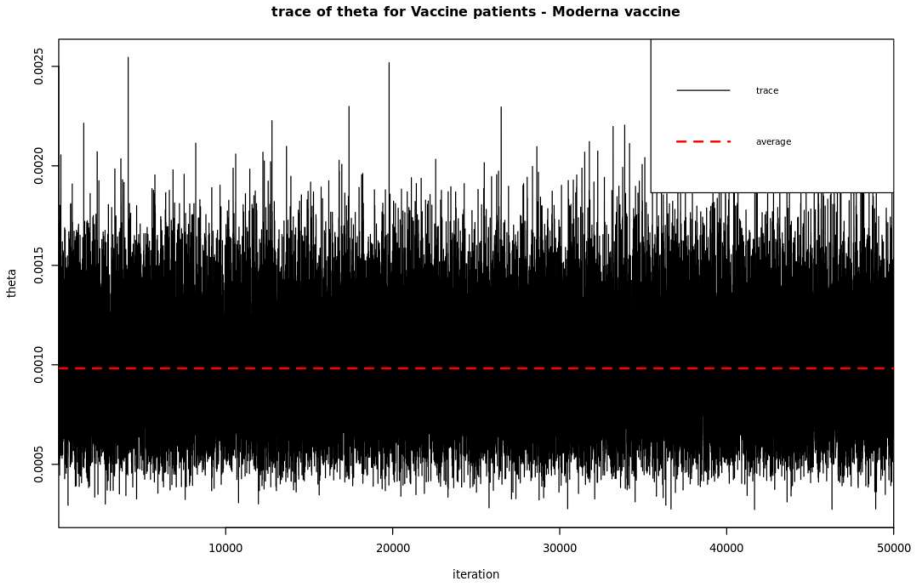
In [12]:
```
my_Vaccine_analysis('Moderna')
```

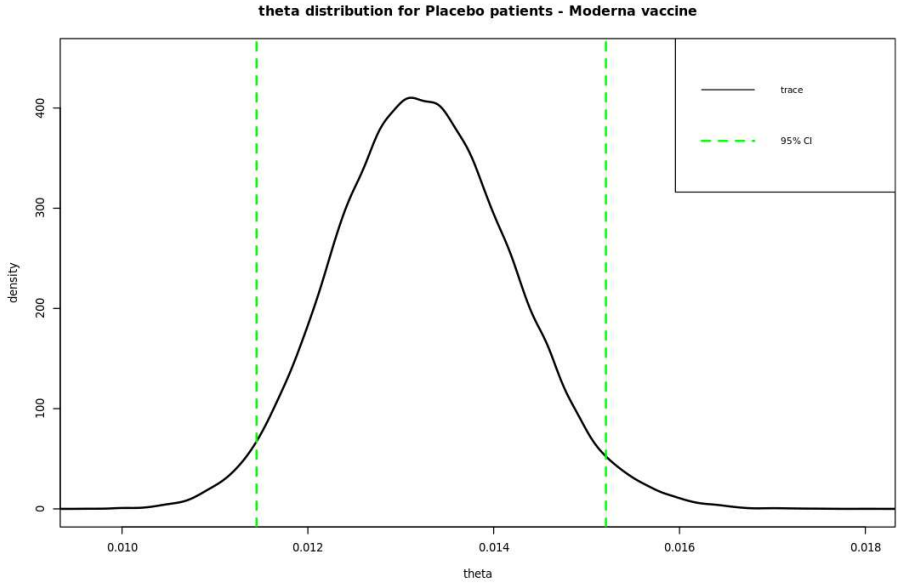
```
           Neg   Pos
  Placebo 13888   185
  Vaccine 14123    11

Iterations = 1:50000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean        SD  Naive SE Time-series SE
theta[1] 0.0132600 0.0009607 4.296e-06      4.296e-06
theta[2] 0.0009825 0.0002630 1.176e-06      1.191e-06

2. Quantiles for each variable:

              2.5%       25%       50%      75%   97.5%
theta[1] 0.0114473 0.0125959 0.0132362 0.013897 0.01521
theta[2] 0.0005354 0.0007946 0.0009595 0.001146 0.00156
```

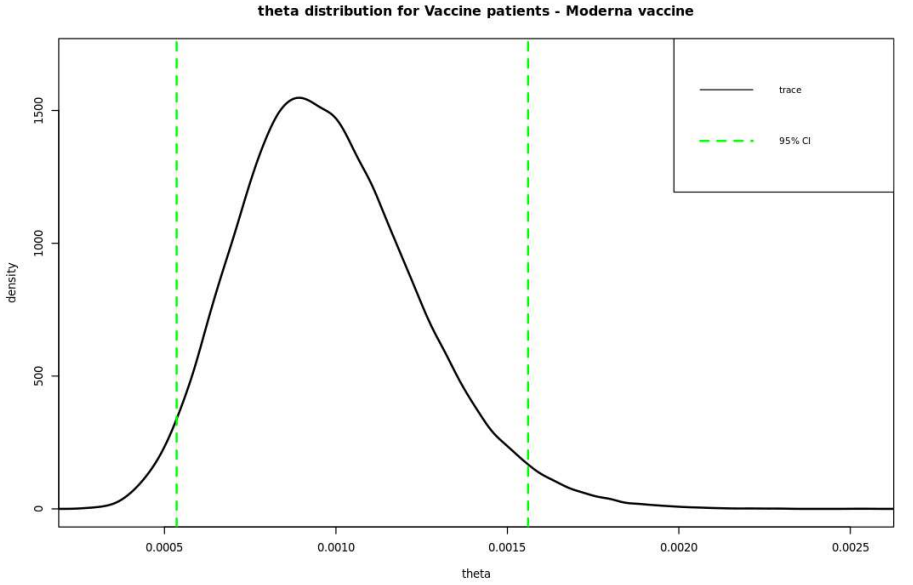
Bayes Factor (Savage-Dickey density ratio)

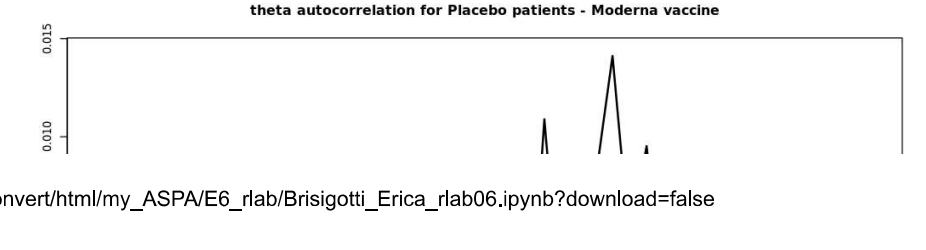
```
BF
--------
3.44e+16

* Evidence Against The Null: 50
```

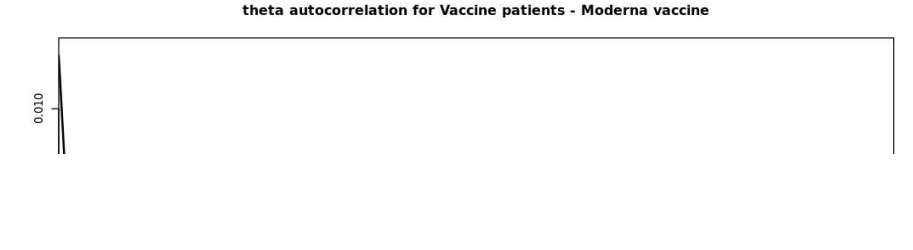

trace of theta for Placebo patients - Moderna vaccine
trace
average
theta
iteration
0.010
0.012
0.014
0.016
0.018
10000
20000
30000
40000
50000

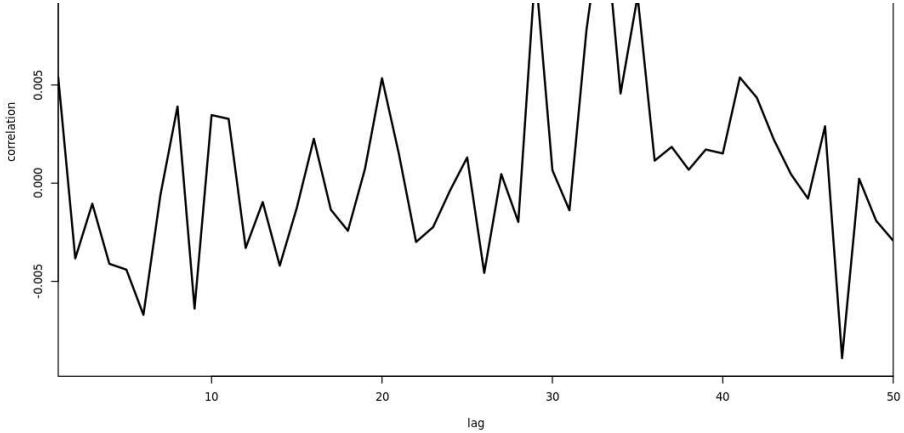

trace of theta for Vaccine patients - Moderna vaccine
trace
average
theta
iteration
0.0005
0.0010
0.0015
0.0020
0.0025
10000
20000
30000
40000
50000

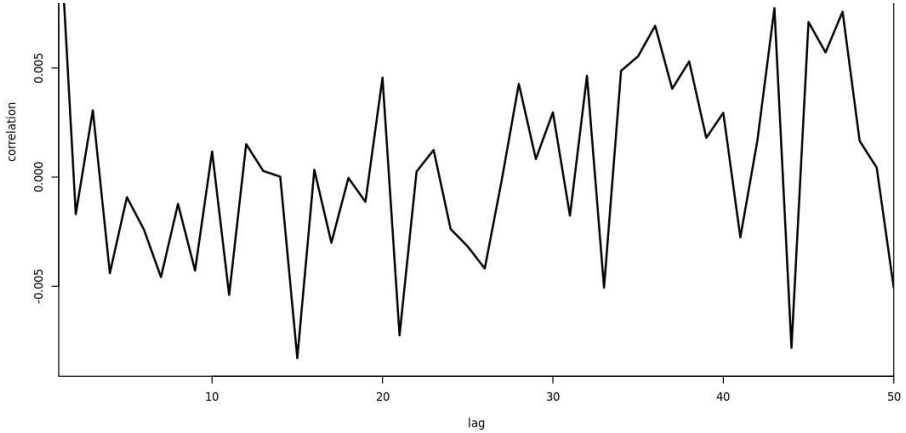

theta distribution for Placebo patients - Moderna vaccine
trace
95% CI
density
theta
0
100
200
300
400
0.010
0.012
0.014
0.016
0.018


theta distribution for Vaccine patients - Moderna vaccine
trace
95% CI
density
theta
0
500
1000
1500
0.0005
0.0010
0.0015
0.0020
0.0025


theta autocorrelation for Placebo patients - Moderna vaccine
0.010
0.015


theta autocorrelation for Vaccine patients - Moderna vaccine
0.010





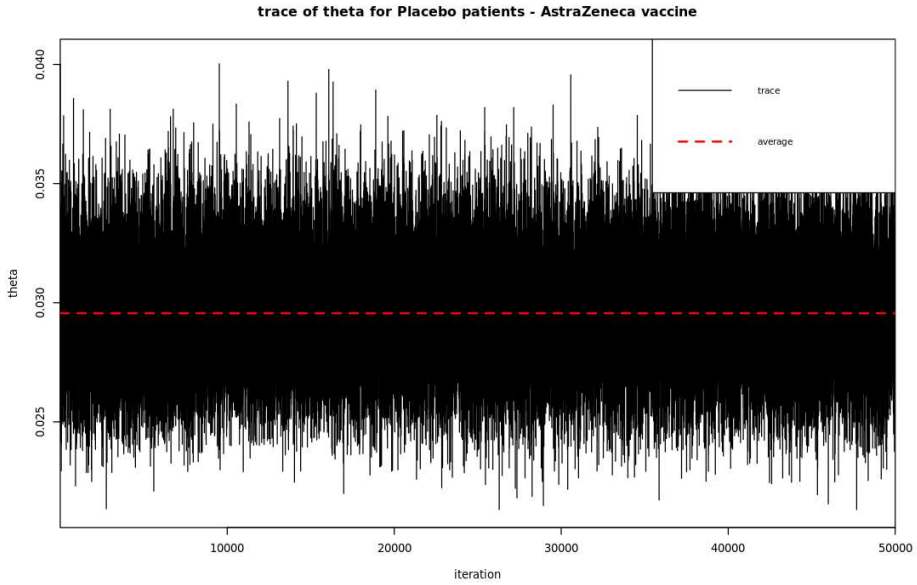
The Bayes Factor is high enough to support the interpretation that "you are less likely to get COVID-19 if you have taken the Moderna vaccine".

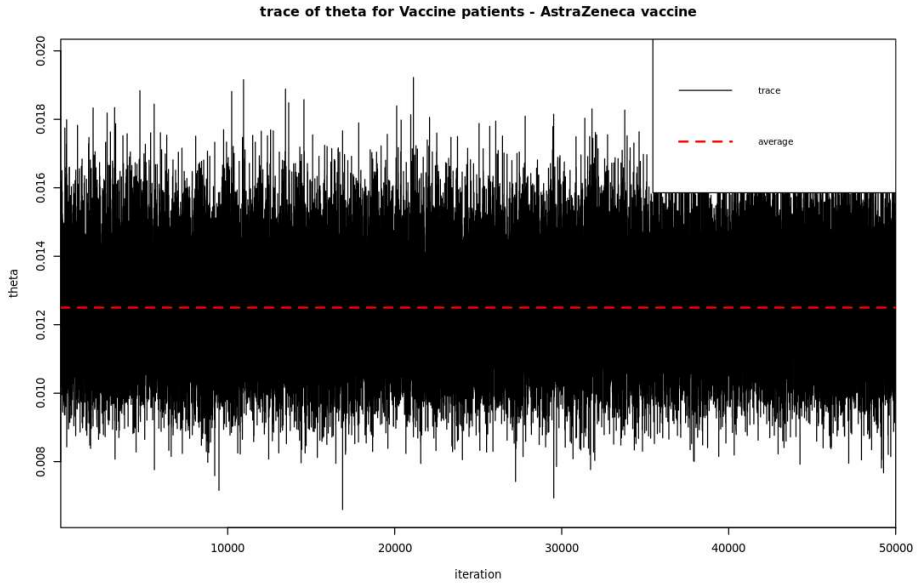
In [13]:
```
my_Vaccine_analysis('AstraZeneca')
```

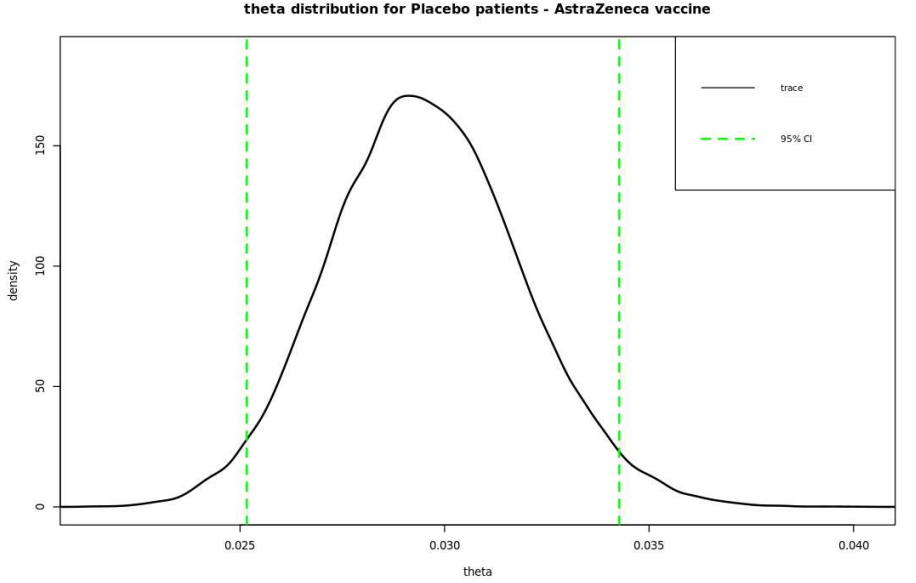
```
          Neg  Pos
  Placebo 5056  154
  Vaccine 5194   64

Iterations = 1:50000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

            Mean       SD  Naive SE Time-series SE
theta[1] 0.02956 0.002325 1.040e-05      1.066e-05
theta[2] 0.01250 0.001513 6.766e-06      6.766e-06

2. Quantiles for each variable:

             2.5%     25%     50%     75%   97.5%
theta[1] 0.025165 0.02795 0.02949 0.03109 0.03427
theta[2] 0.009714 0.01145 0.01243 0.01348 0.01564
```

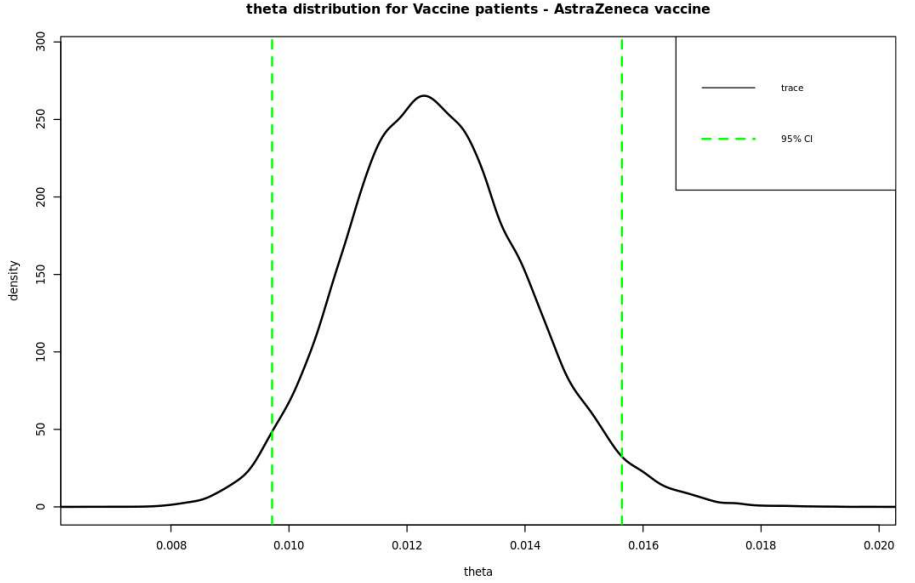
Bayes Factor (Savage-Dickey density ratio)

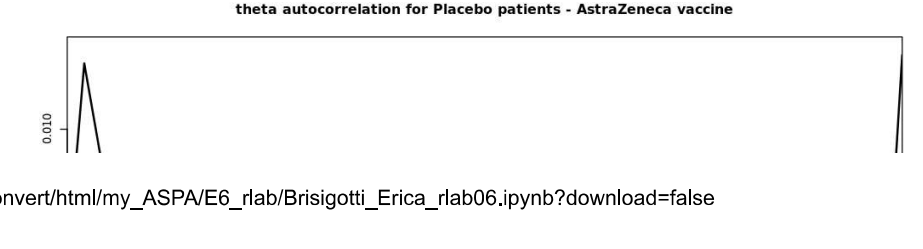
```
BF
-----
0.958

* Evidence Against The Null: 50
```

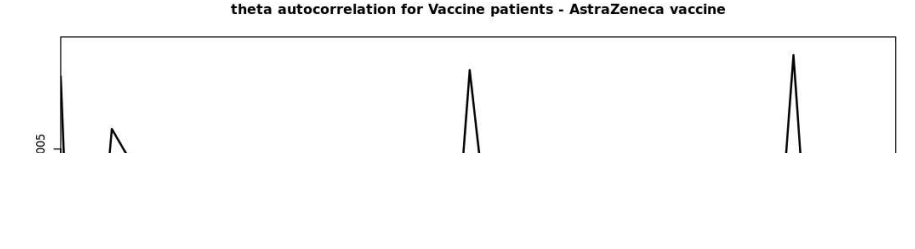

trace of theta for Placebo patients - AstraZeneca vaccine
trace
average
theta
iteration
0.040
0.035
0.030
0.025
10000
20000
30000
40000
50000

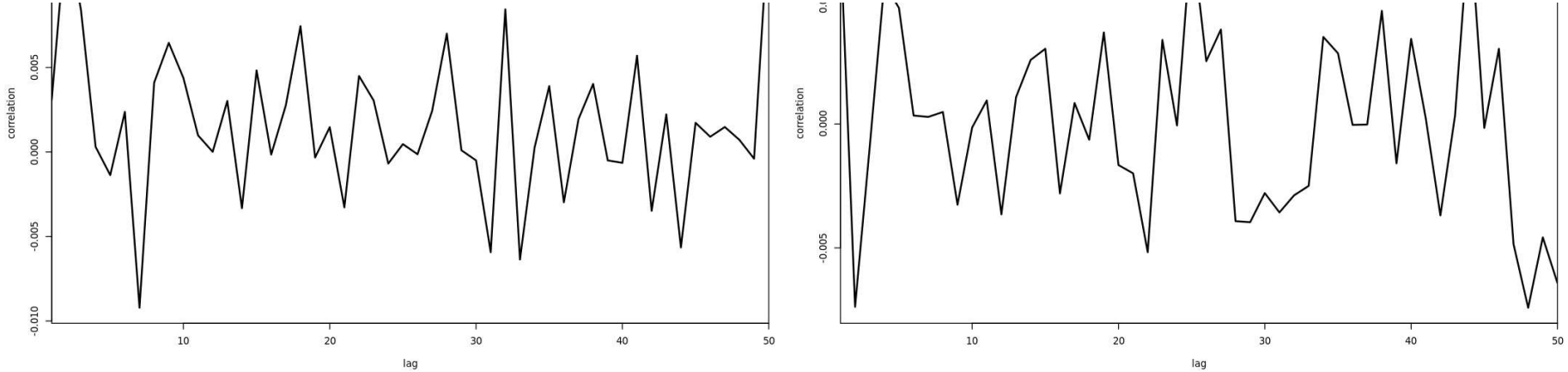

trace of theta for Vaccine patients - AstraZeneca vaccine
trace
average
theta
iteration
0.020
0.018
0.016
0.014
0.012
0.010
0.008
10000
20000
30000
40000
50000


theta distribution for Placebo patients - AstraZeneca vaccine
trace
95% CI
density
theta
150
100
50
0
0.025
0.030
0.035
0.040


theta distribution for Vaccine patients - AstraZeneca vaccine
trace
95% CI
density
theta
300
250
200
150
100
50
0
0.008
0.010
0.012
0.014
0.016
0.018
0.020


theta autocorrelation for Placebo patients - AstraZeneca vaccine
0.010


theta autocorrelation for Vaccine patients - AstraZeneca vaccine



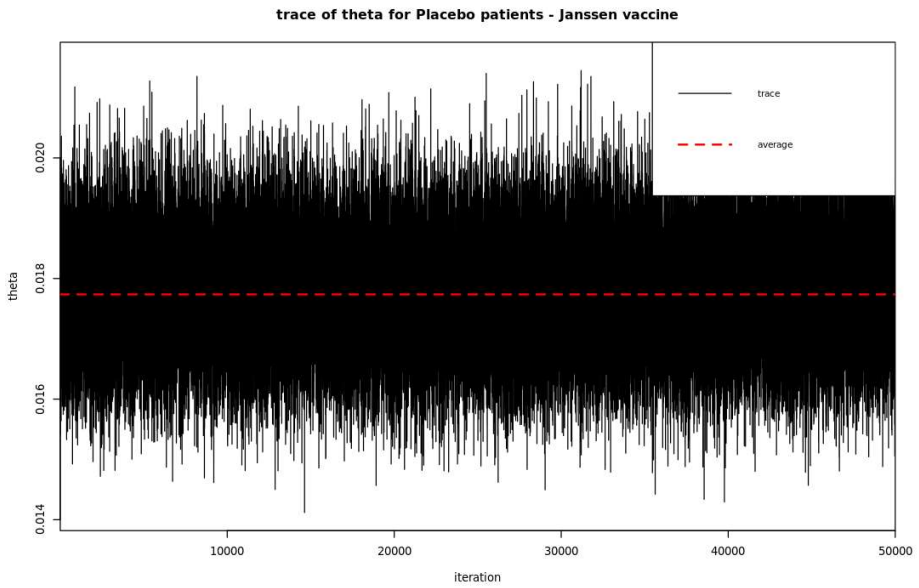
The Bayes Factor doesn't support the rejection of the null hypothesis, so we **cannot** say that "you are less likely to get COVID-19 if you have taken the AstraZeneca vaccine".

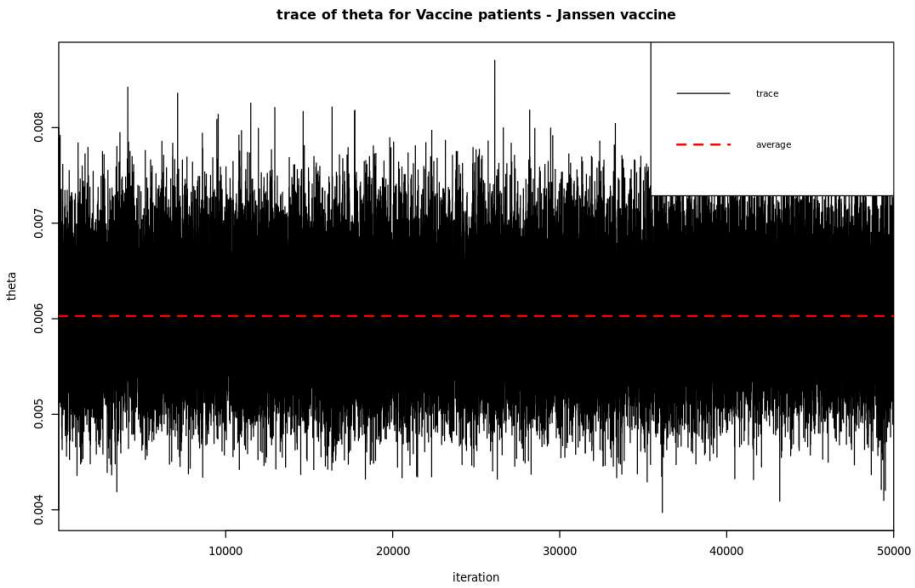
In [14]:
```
my_Vaccine_analysis('Janssen')
```

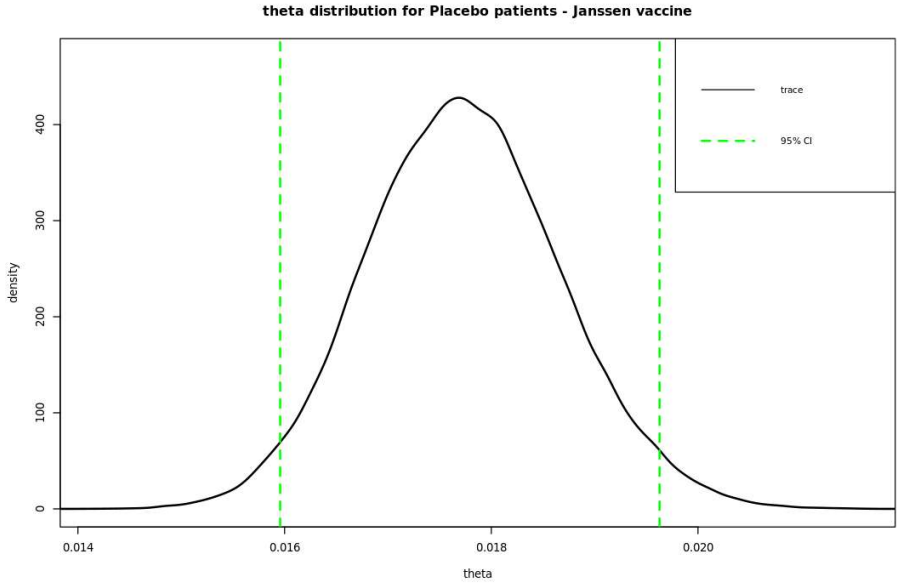
```
            Neg   Pos
  Placebo 19343   348
  Vaccine 19514   116

Iterations = 1:50000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean        SD  Naive SE Time-series SE
theta[1] 0.017737 0.0009351 4.182e-06      4.182e-06
theta[2] 0.006028 0.0005491 2.456e-06      2.522e-06

2. Quantiles for each variable:

             2.5%      25%      50%      75%    97.5%
theta[1] 0.015957 0.017094 0.017719 0.018353 0.019628
theta[2] 0.005005 0.005651 0.006008 0.006386 0.007157
```

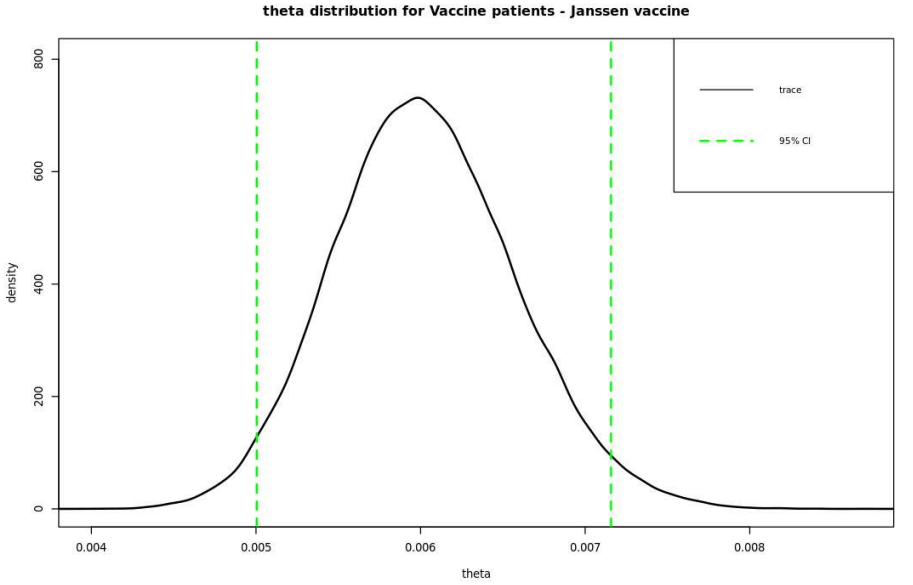
Bayes Factor (Savage-Dickey density ratio)

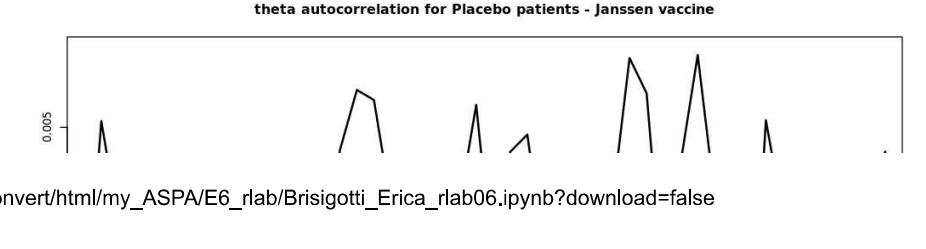
```
BF
------
260.49

* Evidence Against The Null: 50
```

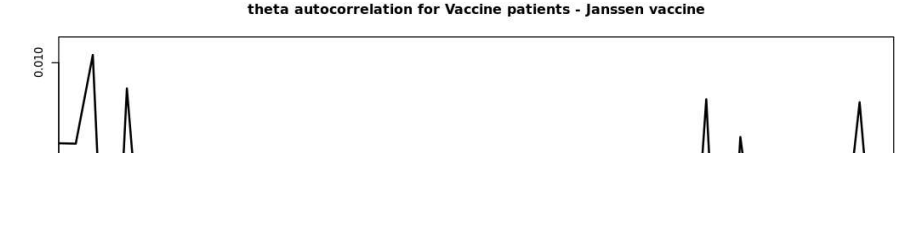

trace of theta for Placebo patients - Janssen vaccine
trace
average
theta
iteration
0.014
0.016
0.018
0.020
10000
20000
30000
40000
50000

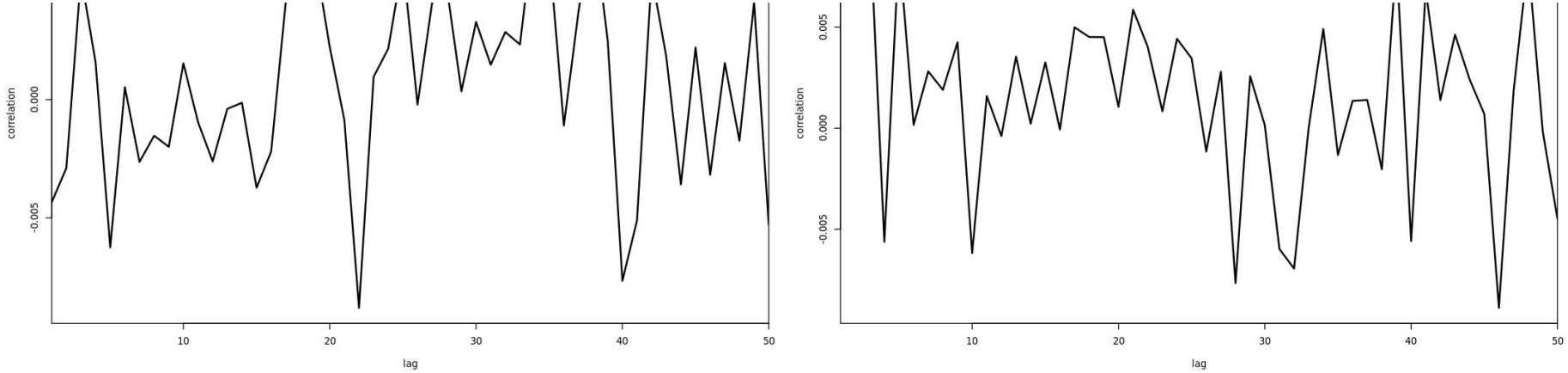

trace of theta for Vaccine patients - Janssen vaccine
trace
average
theta
iteration
0.004
0.005
0.006
0.007
0.008
10000
20000
30000
40000
50000


theta distribution for Placebo patients - Janssen vaccine
trace
95% CI
density
theta
0
100
200
300
400
0.014
0.016
0.018
0.020


theta distribution for Vaccine patients - Janssen vaccine
trace
95% CI
density
theta
0
200
400
600
800
0.004
0.005
0.006
0.007
0.008


theta autocorrelation for Placebo patients - Janssen vaccine
0.005


theta autocorrelation for Vaccine patients - Janssen vaccine
0.010



The Bayes Factor is high enough to support the interpretation that "you are less likely to get COVID-19 if you have taken the Janssen/Jcovden vaccine".

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. A global vaccination dataset is available [5]. The European Centre for Disease Prevention and Control published a downloadable file [6] containing information on COVID-19 vaccination in the EU/EEA.

Analyze the data and produce the following plots:

- number of vaccinated people (cumulative, daily and week average)

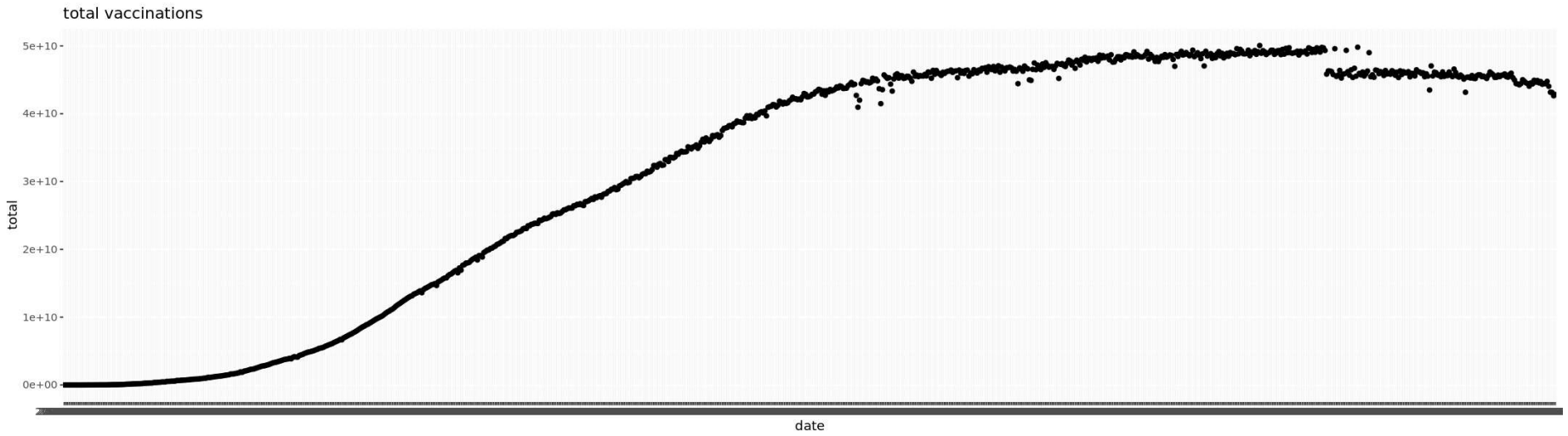
In [15]:
```
library('tidyverse')
```

In [16]:
```
# I upload the data
df = read.csv( 'vaccinations.csv')
head(df)
```

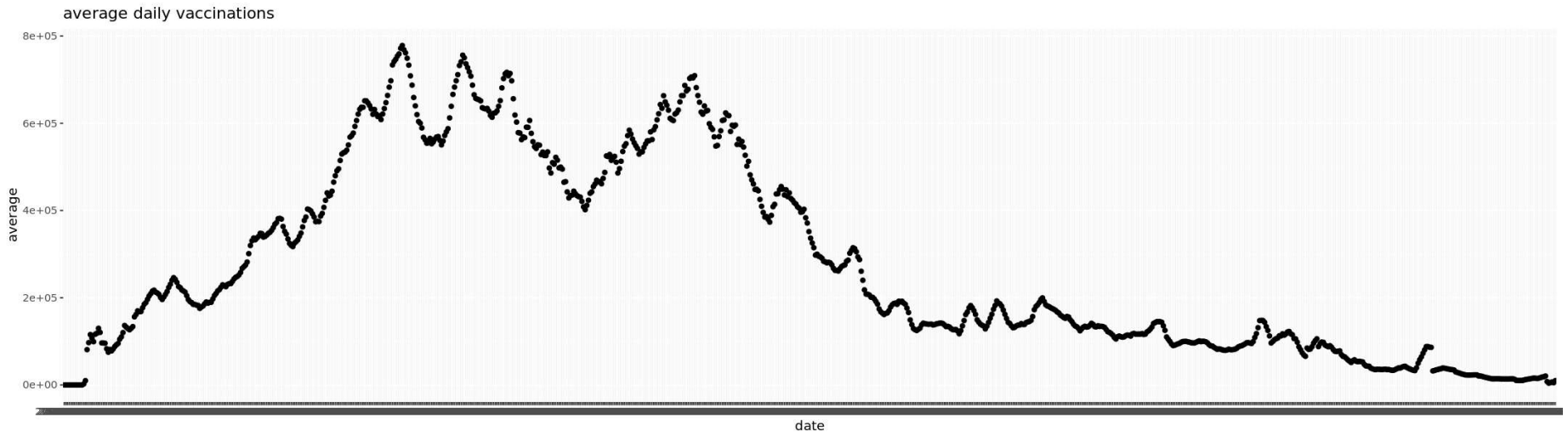
	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	total_boosters	daily_vaccinations_raw	daily_
	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	
1	Afghanistan	AFG	2021-02-22	0	0	NA	NA	NA	
2	Afghanistan	AFG	2021-02-23	NA	NA	NA	NA	NA	
3	Afghanistan	AFG	2021-02-24	NA	NA	NA	NA	NA	
4	Afghanistan	AFG	2021-02-25	NA	NA	NA	NA	NA	
5	Afghanistan	AFG	2021-02-26	NA	NA	NA	NA	NA	
6	Afghanistan	AFG	2021-02-27	NA	NA	NA	NA	NA	

In [17]:
```
# I group the data on a daily basis and save the daily count and average
df %>% group_by(date) %>% summarise( total=sum(total_vaccinations, na.rm=TRUE),
                                     average=mean(daily_vaccinations, na.rm=TRUE) ) -> daily_df
```

In [18]:
```
# I plot the results
# for the daily cumulative sum
options(repr.plot.width=18, repr.plot.height=7)
ggplot(daily_df, aes(date, total)) + geom_point() + theme(aspect.ratio = 1/4) + ggtitle('total vaccinations')
```

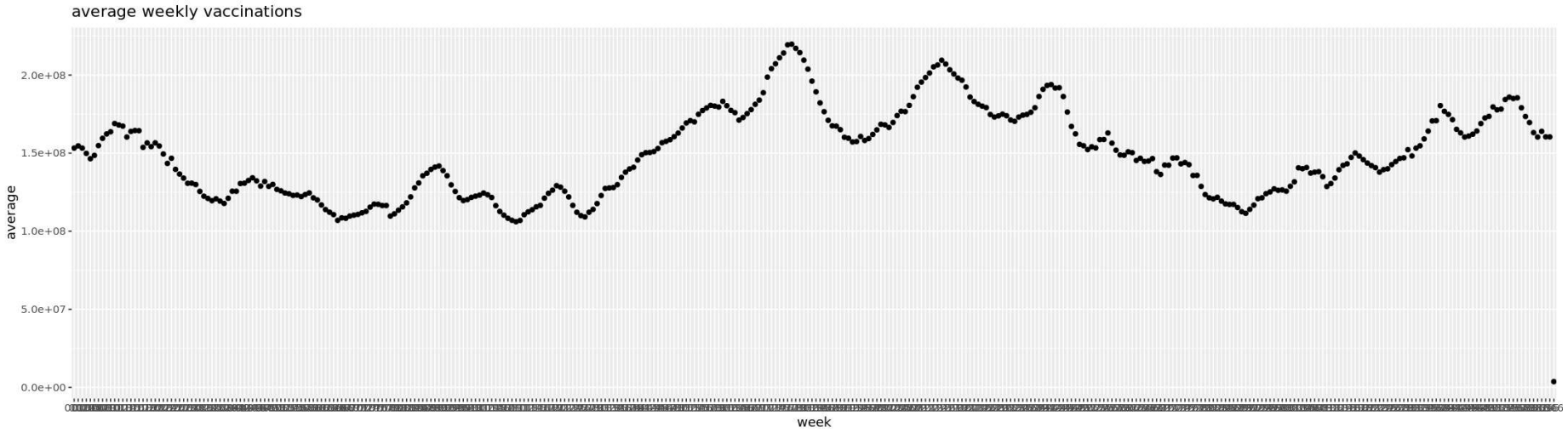


In [19]:
```
# for the daily average
options(repr.plot.width=18, repr.plot.height=7)
ggplot(daily_df, aes(date, average)) + geom_point() + theme(aspect.ratio = 1/4) + ggtitle('average daily vaccinations')
```



In [20]:
```
# I create a column with the number of the week
df["week"] <- strftime( df$date, format="%j" )
# so that I can group the data on a weekly basis and save
df %>% group_by( week ) %>% summarise( average = sum(daily_vaccinations, na.rm=TRUE) ) -> weekly_df
```

In [21]:
```
# for the daily average
options(repr.plot.width=18, repr.plot.height=7)
ggplot(weekly_df, aes(week, average)) + geom_point() + theme(aspect.ratio = 1/4) + ggtitle('average weekly vaccinations')
```



- number of confirmed deaths by COVID-19, both cumulative and weekly average

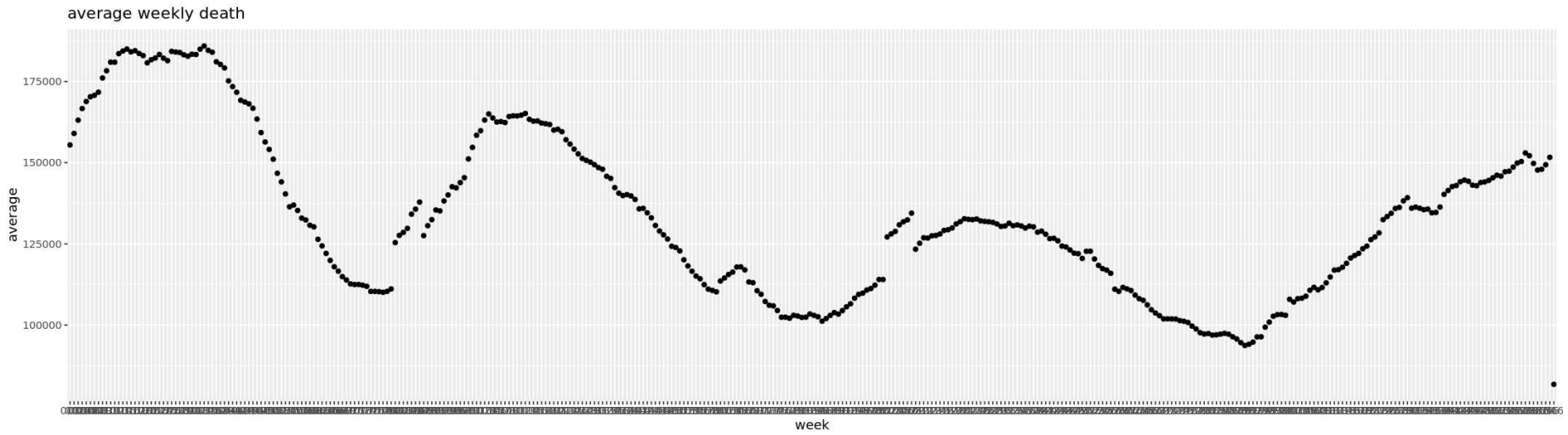
In [22]:
```
dataframe <- read.csv( 'weekly_deaths.csv')
head(dataframe)
```

A data.frame: 6 × 247

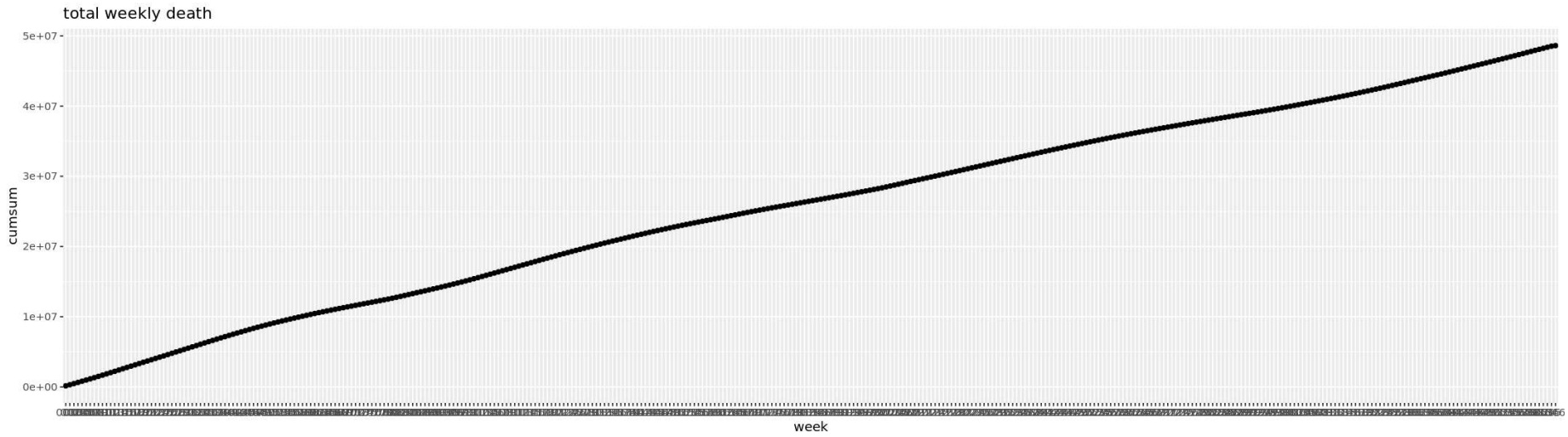
	date	World	Afghanistan	Africa	Albania	Algeria	American.Samoa	Andorra	Angola	Anguilla	⋯	Uruguay	Uzbekistan	Vanuatu
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	⋯	<dbl>	<dbl>	<dbl>
1	2020-01-03	NA	NA	NA	NA	NA	NA	NA	NA	NA	⋯	NA	NA	NA
2	2020-01-04	NA	NA	NA	NA	NA	NA	NA	NA	NA	⋯	NA	NA	NA
3	2020-01-05	NA	NA	NA	NA	NA	NA	NA	NA	NA	⋯	NA	NA	NA
4	2020-01-06	NA	NA	NA	NA	NA	NA	NA	NA	NA	⋯	NA	NA	NA
5	2020-01-07	NA	NA	NA	NA	NA	NA	NA	NA	NA	⋯	NA	NA	NA
6	2020-01-08	0	0	0	0	0	0	0	0	0	⋯	0	0	0

In [23]:
```
# I create a column with the number of the week
dataframe["week"] <- strftime( dataframe$date, format="%j" )
# so that I can group the data on a weekly basis and save
dataframe %>% group_by( week ) %>% summarise( sum=sum(World, na.rm=TRUE), average=sum(World, na.rm=TRUE) ) -> weekly_datafram
weekly_dataframe['cumsum'] <- cumsum(weekly_dataframe['sum'])
```

In [24]:
```
# I plot the total deaths by week
options(repr.plot.width=18, repr.plot.height=7)
ggplot(weekly_dataframe, aes(week, average)) + geom_point() + theme(aspect.ratio = 1/4) + ggtitle('average weekly death')
```



In [25]:
```
# I plot the total deaths by week
options(repr.plot.width=18, repr.plot.height=7)
ggplot(weekly_dataframe, aes(week, cumsum)) + geom_point() + theme(aspect.ratio = 1/4) + ggtitle('total weekly death')
```



References

[1] https://www.ema.europa.eu/en/documents/overview/covid-19-vaccine-janssen-epar-medicine-overview_en.pdf

[2] https://www.ema.europa.eu/en/documents/overview/spikevax-previously-covid-19-vaccine-moderna-epar-medicine-overview_en.pdf

[3] https://www.ema.europa.eu/en/documents/overview/vaxzevria-previously-covid-19-vaccine-astrazeneca-epar-medicine-overview_en.pdf

[4] https://www.ema.europa.eu/en/documents/overview/jcovden-previously-covid-19-vaccine-janssen-epar-medicine-overview_en.pdf

[5] https://ourworldindata.org/covid-vaccinations

[6] https://github.com/owid/covid-19-data/tree/master/public/data

In [ ]:

In [ ]: