Assign_1

Franco Meng

2024-09-01

```
library(ggplot2)
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.6 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using ‘reduce_sum()‘ or ‘map_rect()‘ Stan functions,
## change ‘threads_per_chain‘ option:
## rstan_options(threads_per_chain = 1)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
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
```

```
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

TASK 1

```r
y.freq <- c(71, 28, 5, 2, 2, 1)  ## Generate data point by using Douglas Firs data. FS distribution.
y.obs <- rep(1:6, times = y.freq)
y.obs
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 4 4 5 5 6
```

```r
sum(y.obs)
```

```
## [1] 166
```

```r
length(y.obs)  #sample size
```

```
## [1] 109
```

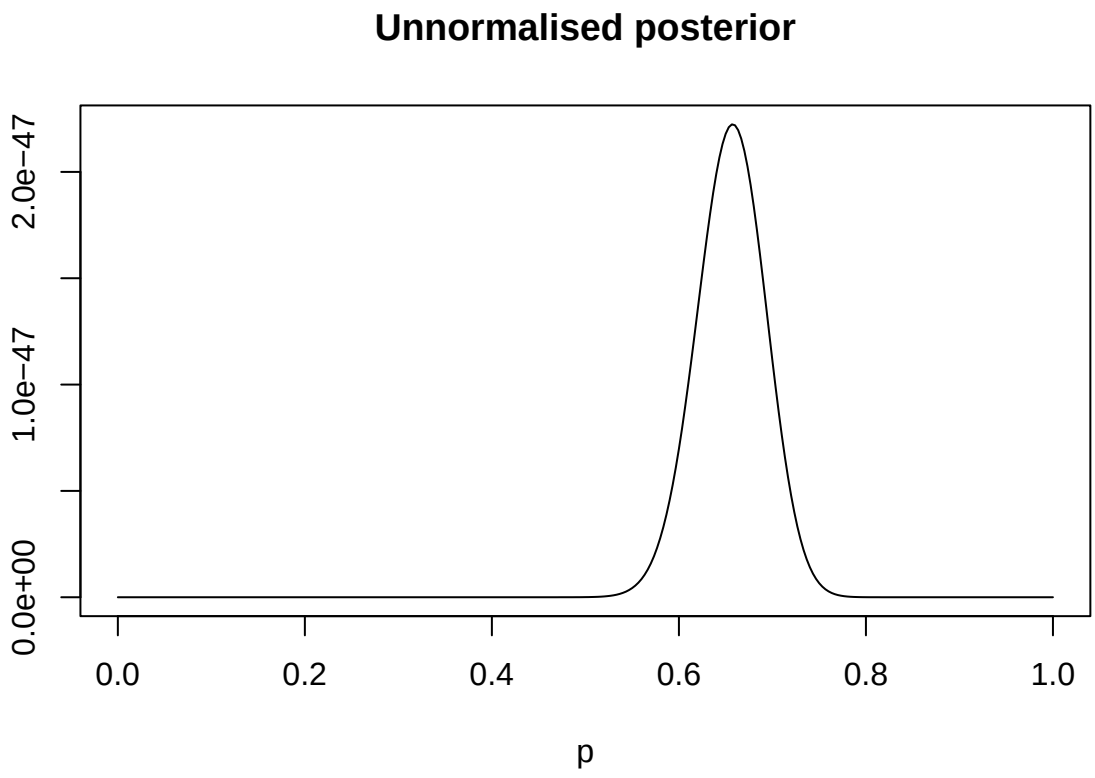
```r
sum(y.obs) - length(y.obs)
```

```
## [1] 57
```

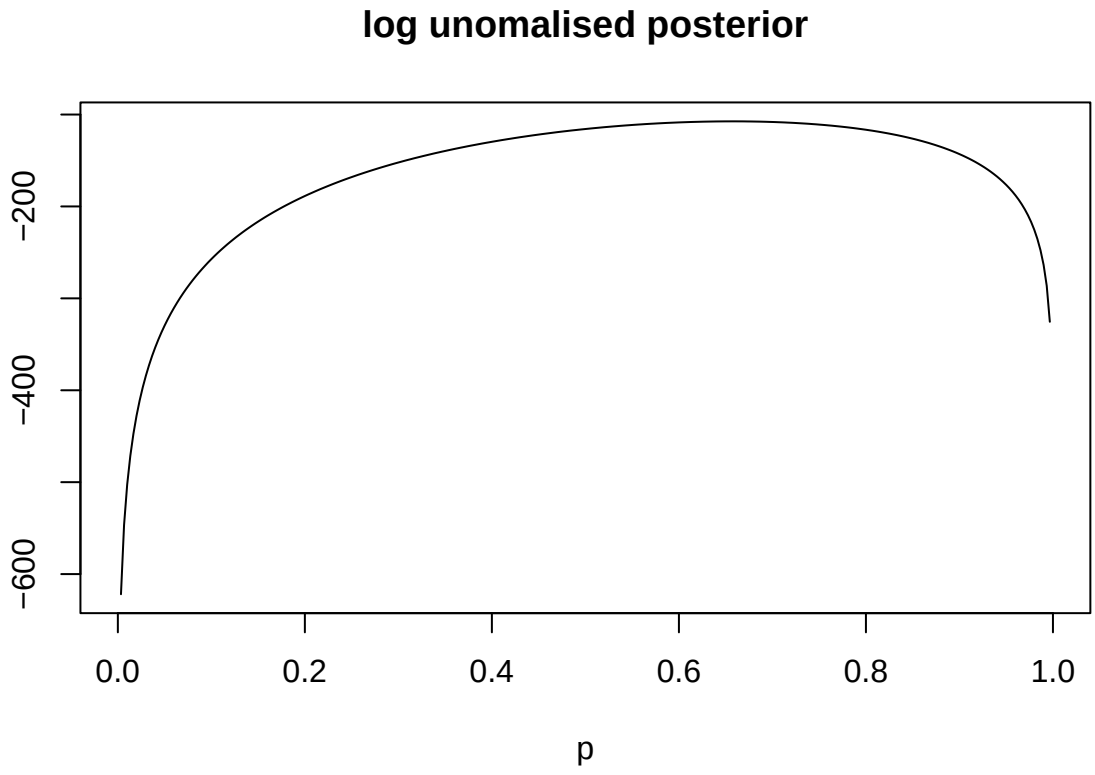
```r
table(y.obs)
```

```
## y.obs
##  1  2  3  4  5  6
## 71 28  5  2  2  1
```

```r
posterior <- function(p) p^(109+p) * (1-p)^(58-p) #posterior distribution by using likelihood of FS and
logit <- function(p) log(p/(1-p)) #logit function
invlogit <- function(lo) 1/(1+exp(-lo)) # back to probability
curve(x^(109+x)*(1-x)^(58-x), from=0, to=1, n=301, ylab="", xlab="p",
      main="Unnormalised posterior")
```




```
log.posterior <- function(p) (109+p)*log(p) + (58-p)*log(1-p)
curve(log.posterior(x), from=0, to=1, n=301, ylab="", xlab="p", main="log unomalised posterior")
```




```
# Below is the implementation of random walk MH algorithm, starting the chain on p=0.2 as required.
# Sigma = 0.01, calculate the acceptance rate.


B <- 10000
chain <- rep(0, B+1)
chain[1] <- 0.2
num.accept <- 0
sd <- 0.01
##sd <- 0.15
for(i in 1:B){
  ptm1 <- chain[i]
  xt <- invlogit(logit(ptm1) + rnorm(1,0,sd))
  lapt <- log.posterior(xt) - log.posterior(ptm1) + log(xt*(1-xt)) - log(ptm1*(1-ptm1))
  if( runif(1) <= exp(lapt) ){
    chain[i+1] <- xt
    num.accept <- num.accept + 1
  }else
    chain[i+1] <- ptm1
}
num.accept/B
```

```
## [1] 0.9697
```

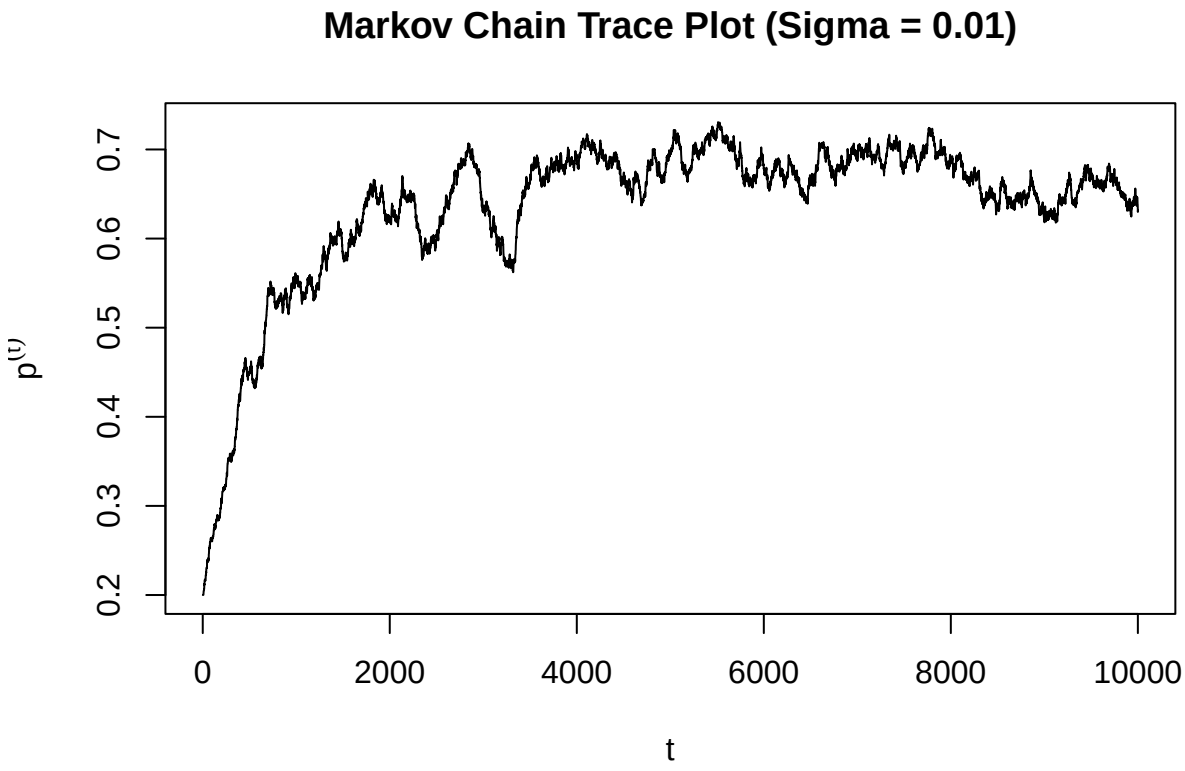
```r
mean(chain)
```

```
## [1] 0.636045
```

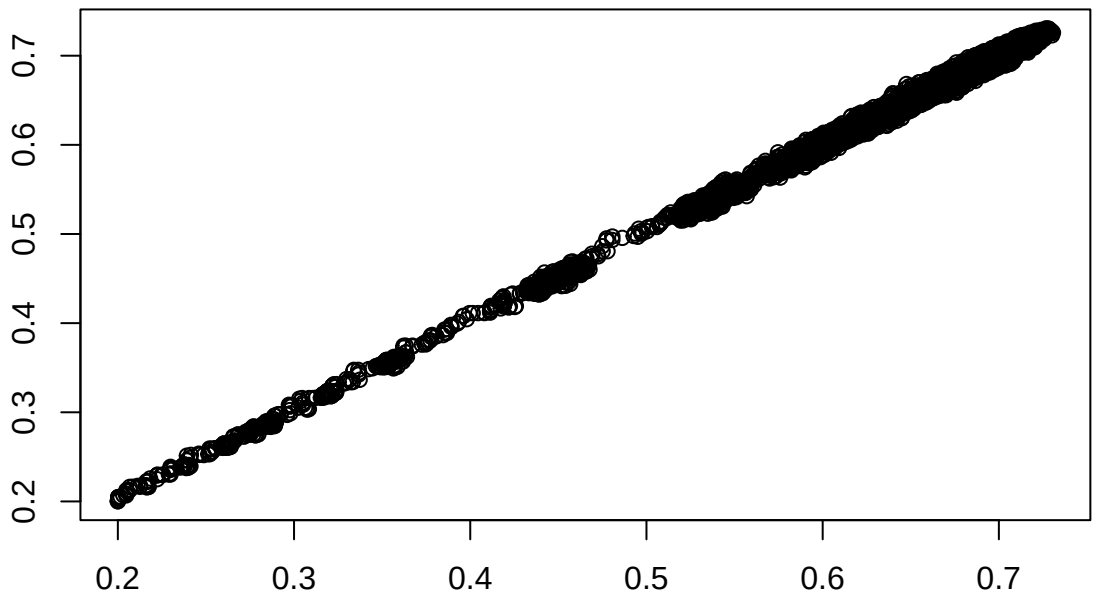
```r
sd(chain)
```

```
## [1] 0.08716853
```

```r
plot(chain, type="l", xlab="t", ylab=expression(p^{(t)}), main="Markov Chain Trace Plot (Sigma = 0.01)")
```




```r
ind <- 1:B
plot(chain[ind], chain[ind+5], xlab="", ylab="", main="Autocorrelation at lag 5 (Sigma = 0.01)")
```




```
## Below is the implementation of random walk MH algorithm, starting the chain on p=0.2 as required.
## Sigma = 10, calculate the acceptance rate.
B <- 10000
chain <- rep(0, B+1)
chain[1] <- 0.2
num.accept <- 0
sd <- 10
for(i in 1:B){
  ptm1 <- chain[i]
  xt <- invlogit(logit(ptm1) + rnorm(1,0,sd))
  lapt <- log.posterior(xt) - log.posterior(ptm1) + log(xt*(1-xt)) - log(ptm1*(1-ptm1))
  if( runif(1) <= exp(lapt) ){
    chain[i+1] <- xt
    num.accept <- num.accept + 1
  }else
    chain[i+1] <- ptm1
}
num.accept/B
```

```
## [1] 0.0239
```

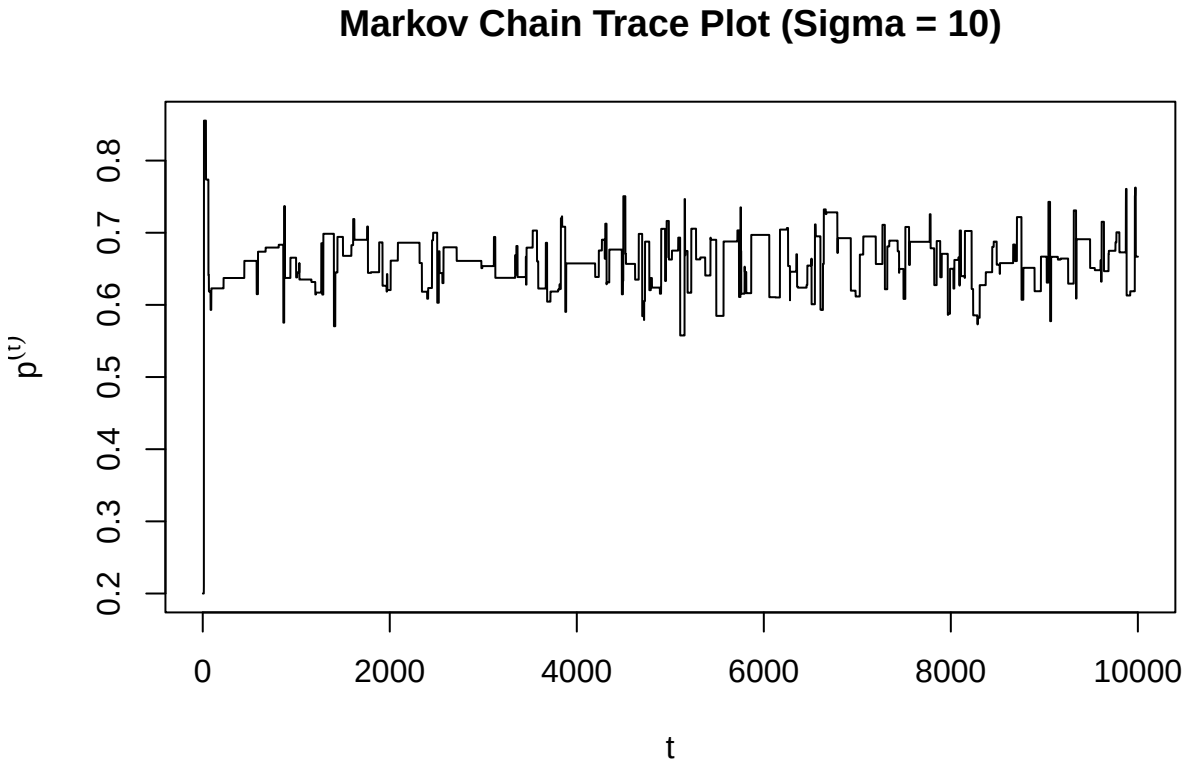
```
mean(chain)
```

```
## [1] 0.660465
```

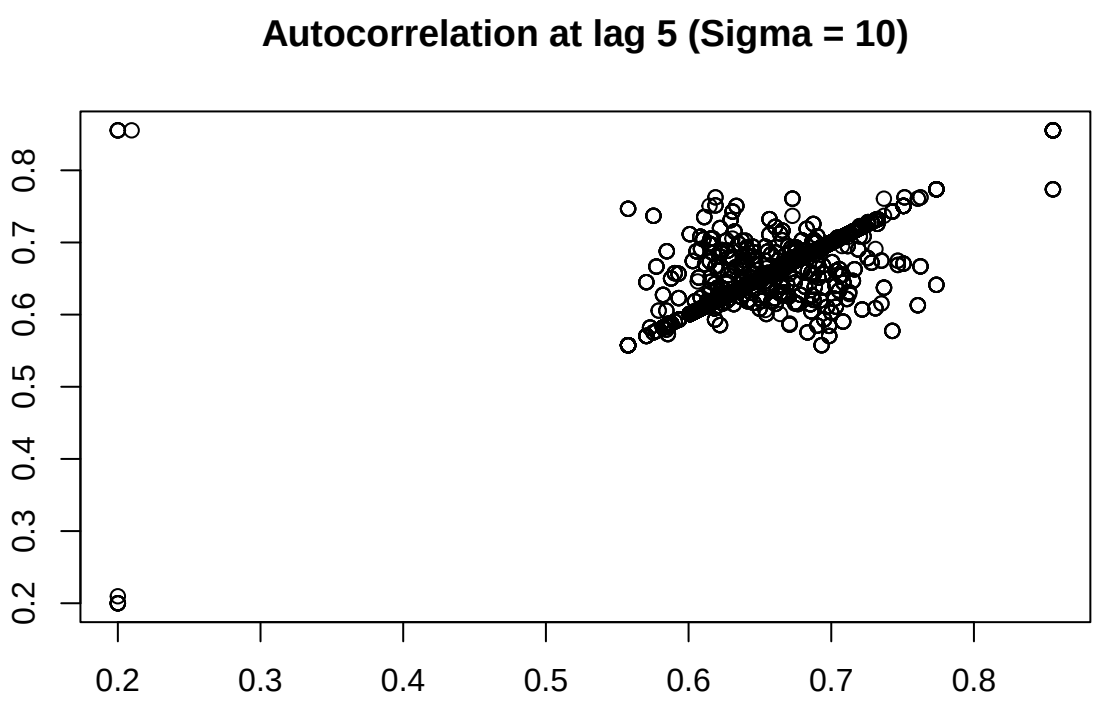
```r
sd(chain)
```

```
## [1] 0.03843056
```

```r
plot(chain, type="l", xlab="t", ylab=expression(p^{(t)}), main="Markov Chain Trace Plot (Sigma = 10)")
```




```r
ind <- 1:B
plot(chain[ind], chain[ind+5], xlab="", ylab="", main="Autocorrelation at lag 5 (Sigma = 10)")
```




```
## Below is the implementation of random walk MH algorithm, starting the chain on p=0.2 as required.
## Sigma = 0.15, calculate the acceptance rate.

B <- 10000
chain <- rep(0, B+1)
chain[1] <- 0.2
num.accept <- 0
sd <- 0.15
for(i in 1:B){
  ptm1 <- chain[i]
  xt <- invlogit(logit(ptm1) + rnorm(1,0,sd))
  lapt <- log.posterior(xt) - log.posterior(ptm1) + log(xt*(1-xt)) - log(ptm1*(1-ptm1))
  if( runif(1) <= exp(lapt) ){
    chain[i+1] <- xt
    num.accept <- num.accept + 1
  }else
    chain[i+1] <- ptm1
}
num.accept/B
```

```
## [1] 0.7398
```

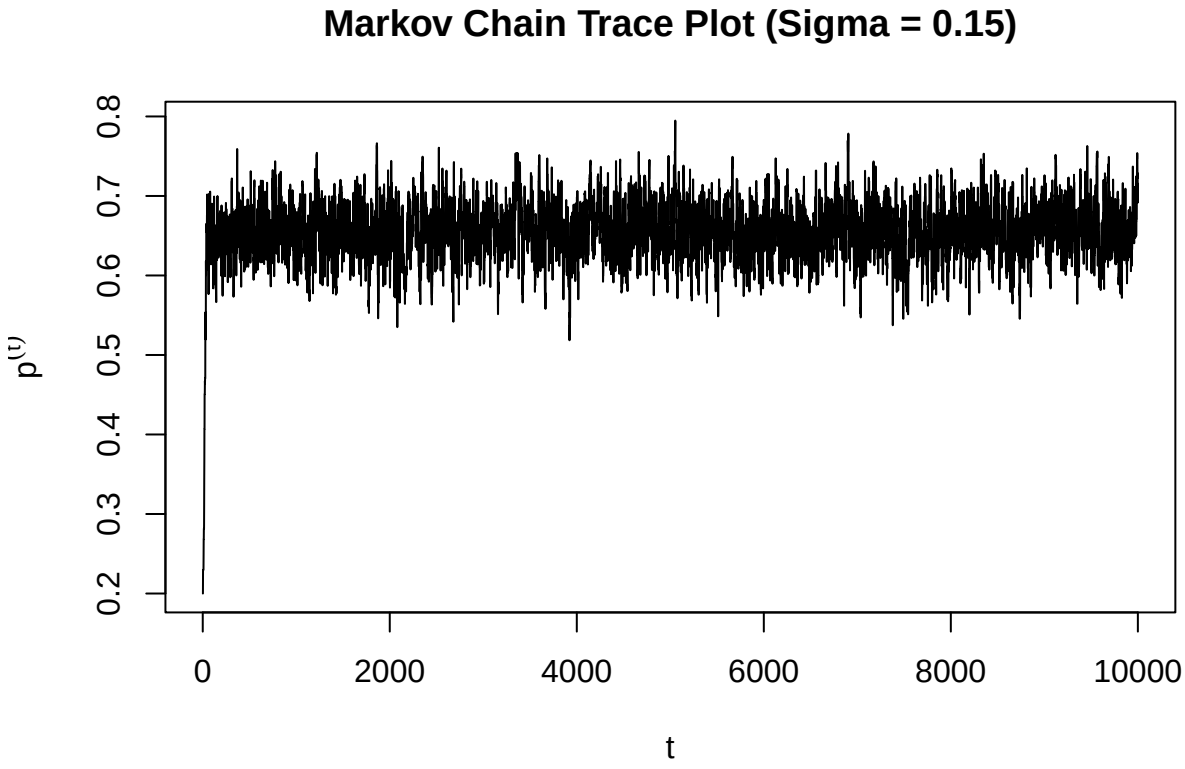
```
mean(chain)
```

```
## [1] 0.6555143
```

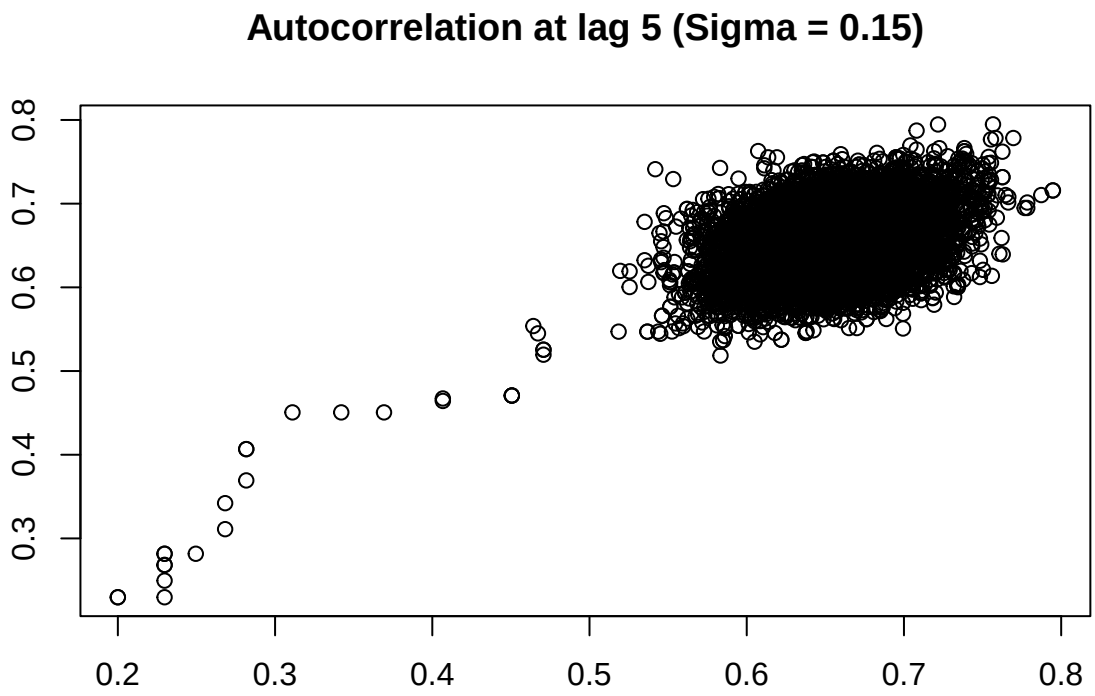
```r
sd(chain)
```

```
## [1] 0.0403966
```

```r
plot(chain, type="l", xlab="t", ylab=expression(p^{(t)}), main="Markov Chain Trace Plot (Sigma = 0.15)")
```

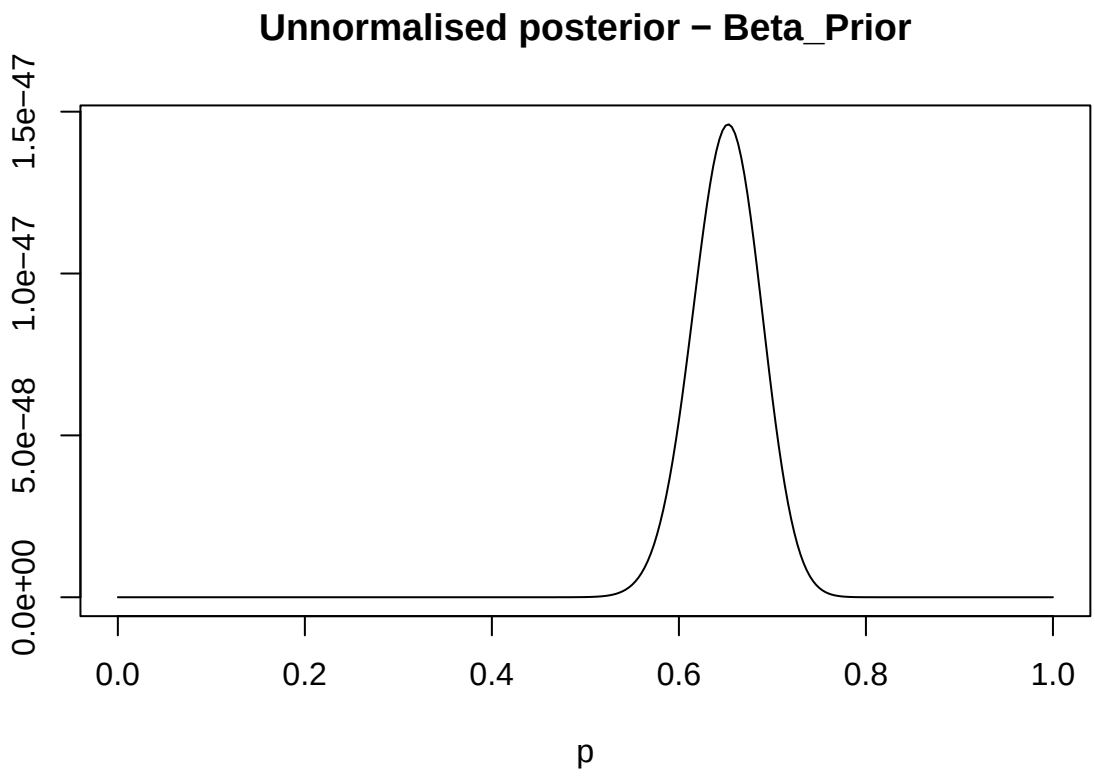



```r
ind <- 1:B
plot(chain[ind], chain[ind+5], xlab="", ylab="", main="Autocorrelation at lag 5 (Sigma = 0.15)")
```

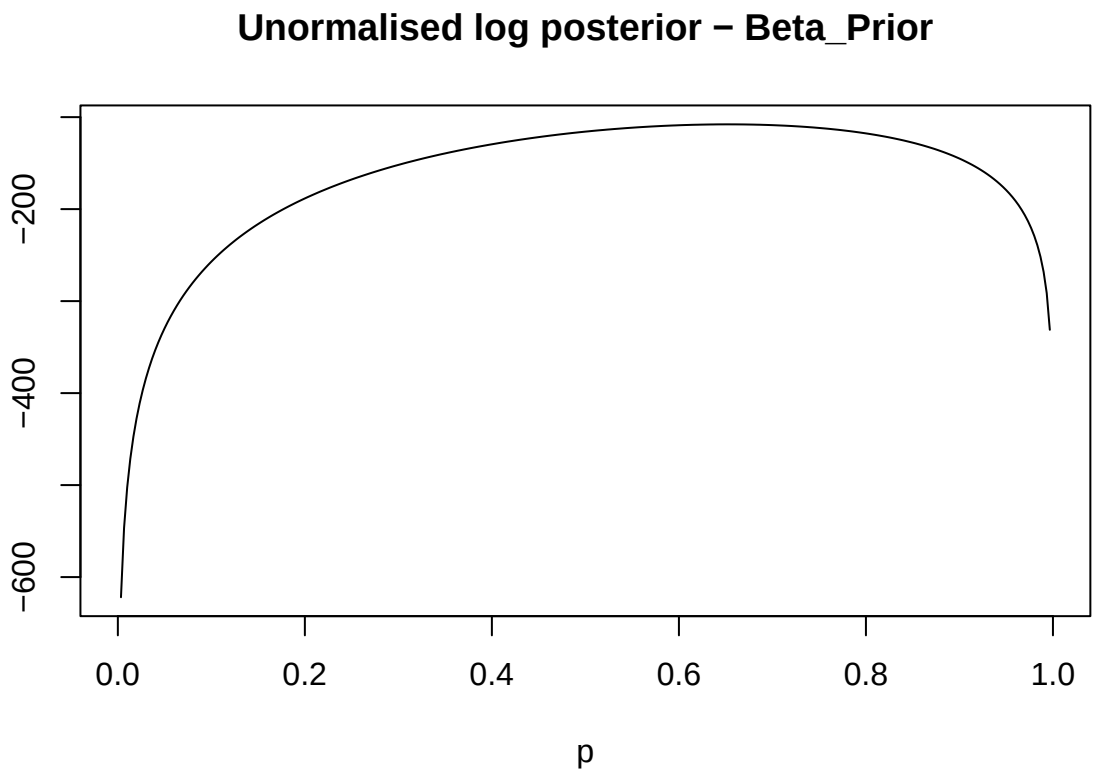


Task 2.

```
alpha <- 1
beta <- 1
posterior_beta <- function(p) p^(108+alpha) * (1-p)^(57+beta)

curve(x^(108+alpha)*(1-x)^(57+beta), from=0, to=1, n=301, ylab="", xlab="p",
      main="Unnormalised posterior - Beta_Prior")
```

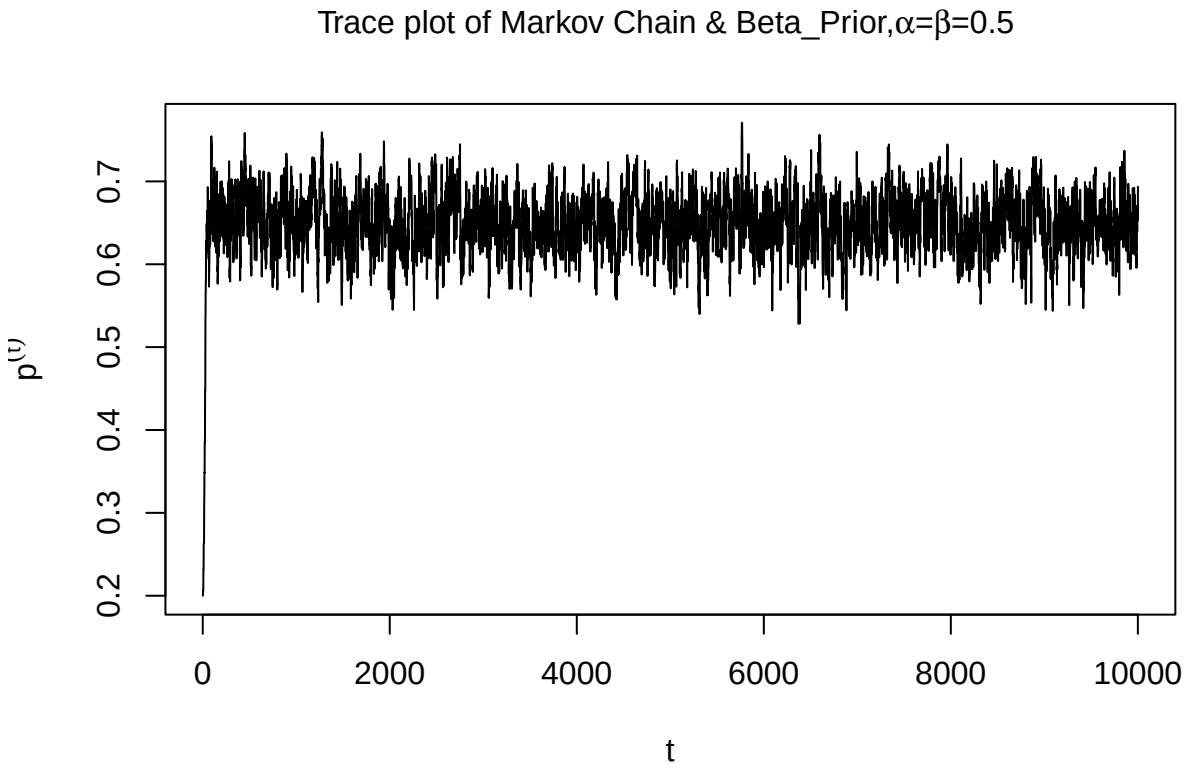



```
log.posterior_beta <- function(p) (108+alpha)*log(p) + (57+beta)*log(1-p)
curve(log.posterior_beta(x), from=0, to=1, n=301, ylab="", xlab="p", main="Unormalised log posterior - B
```

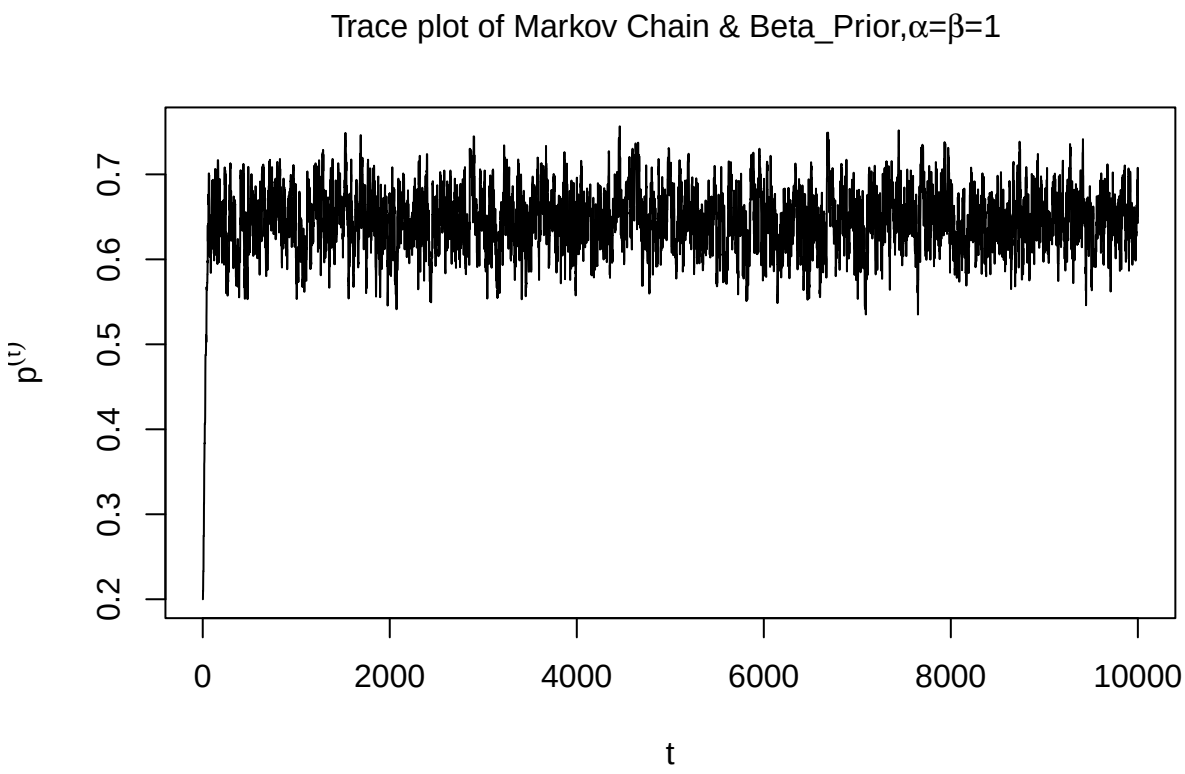



```
## Below showing the trace plot for different alpha and beta.


k_seq <- seq(-1, 18, by = 1)
p_k <- rep(0, length(k_seq))
log_gammak <- rep(0, length(k_seq))
for ( ii in 1:length(k_seq)) {
      alpha <- 2^k_seq[ii]
      beta <- 2^k_seq[ii]
      log.posterior_beta <- function(p) (108+alpha)*log(p) + (57+beta)*log(1-p)
      B <- 10000
      chain <- rep(0, B+1)
      chain[1] <- 0.2
      num.accept <- 0
      sd <- 0.1
      for(i in 1:B){
        ptm1 <- chain[i]
        xt <- invlogit(logit(ptm1) + rnorm(1,0,sd))
        lapt <- log.posterior_beta(xt) - log.posterior_beta(ptm1) + log(xt*(1-xt)) - log(ptm1*(1-ptm1))

        if( runif(1) <= exp(lapt) ){
          chain[i+1] <- xt
          num.accept <- num.accept + 1
        }else
          chain[i+1] <- ptm1
      }
```

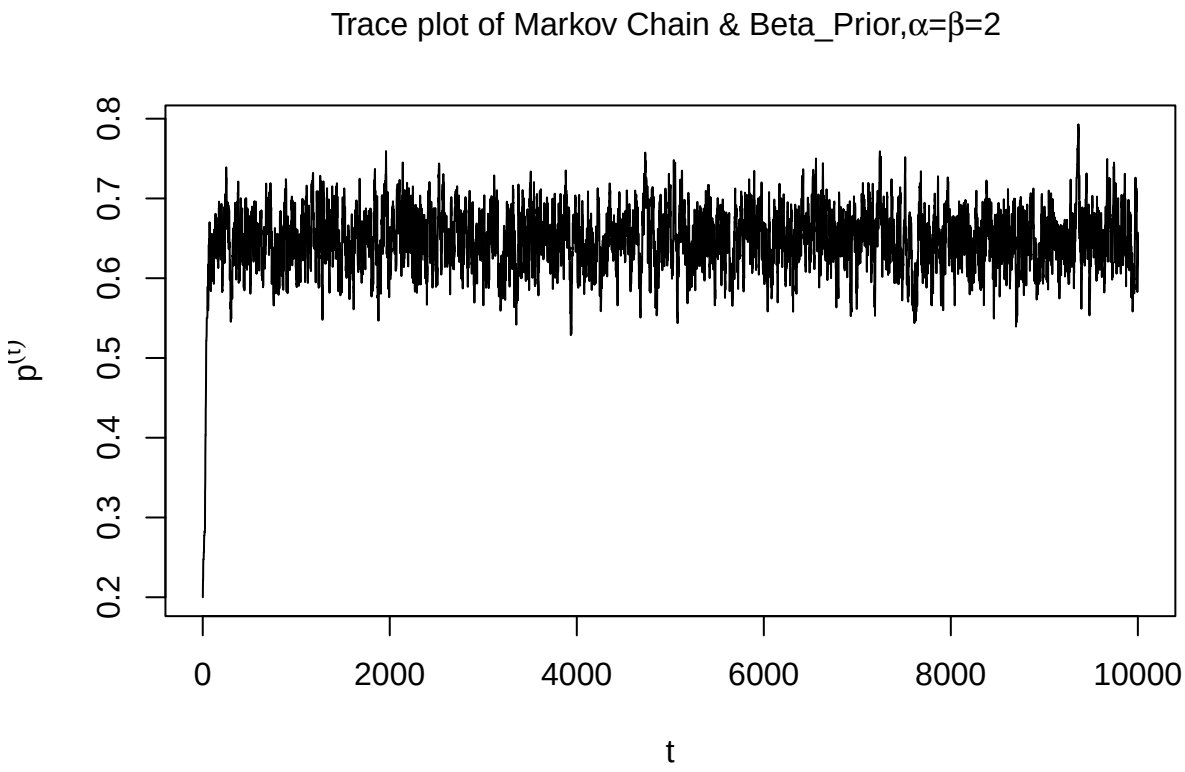
```r
  plot(chain, type="l", xlab="t", ylab=expression(p^{(t)}),main= bquote(paste("Trace plot of Markov Cha
  p_k[ii] <- mean(chain[1000:B])
  print(alpha)
  log_gammak[ii] <- log(alpha)

}
```



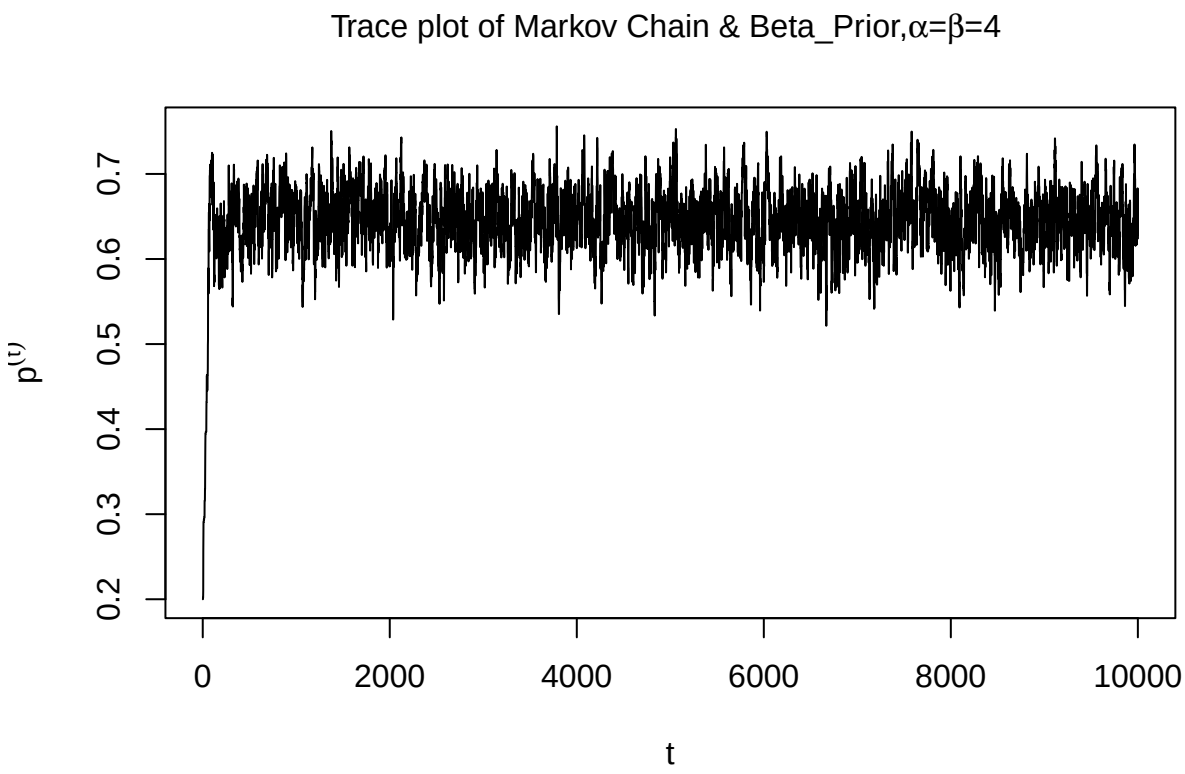

```
## [1] 0.5
```


Trace plot of Markov Chain & Beta_Prior,α=β=1
p(t)
0.2
0.3
0.4
0.5
0.6
0.7
0
2000
4000
6000
8000
10000
t

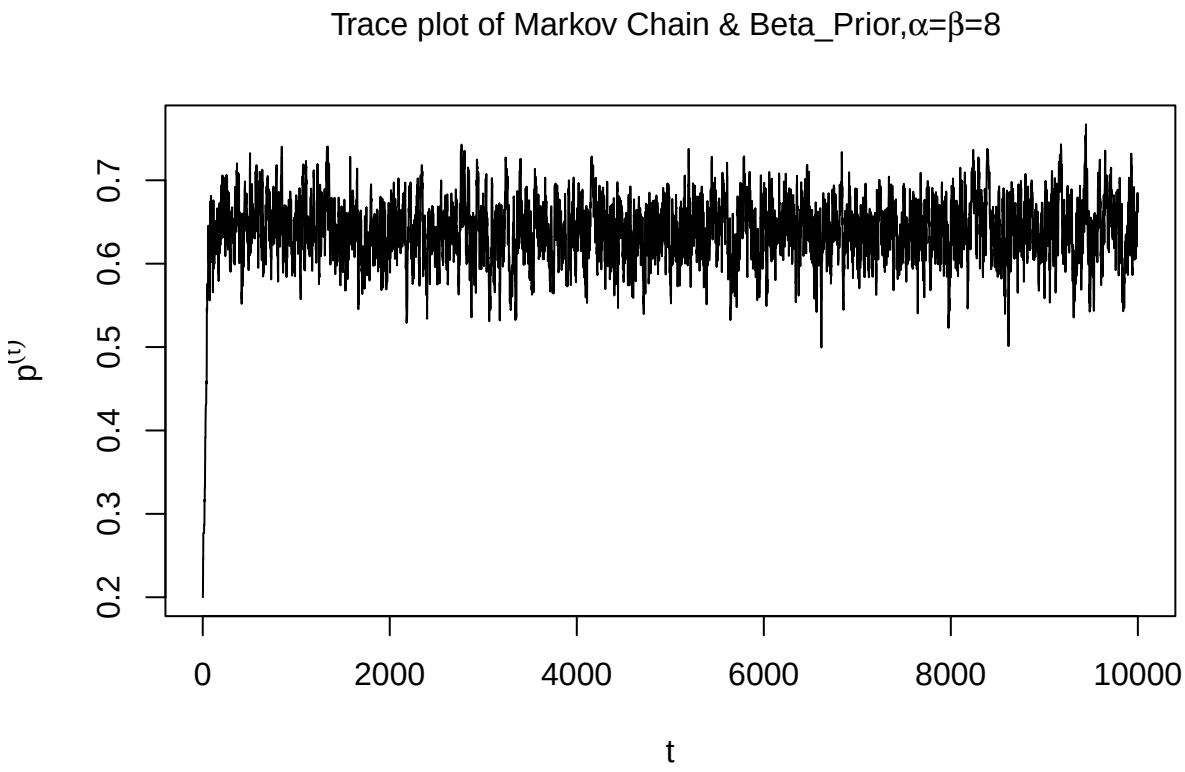
```
## [1] 1
```


Trace plot of Markov Chain & Beta_Prior,α=β=2
p(t)
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
2000
4000
6000
8000
10000
t

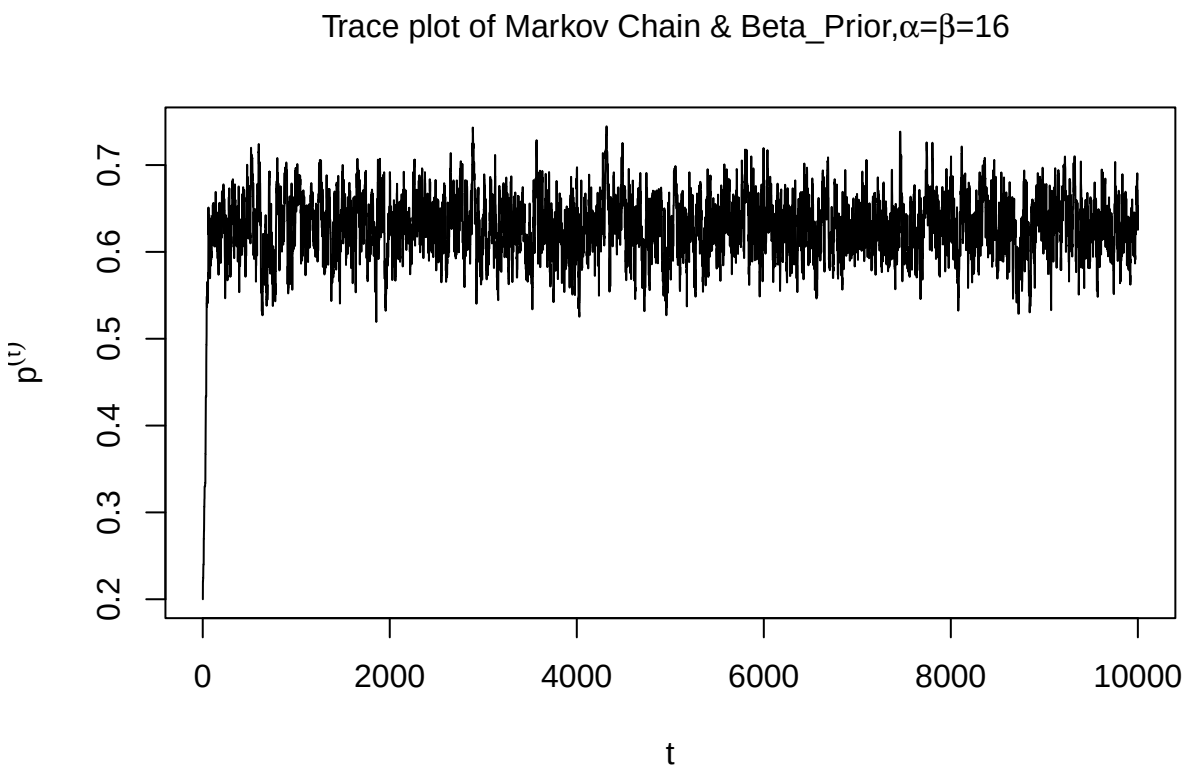
```
## [1] 2
```


Trace plot of Markov Chain & Beta_Prior,α=β=4
p(t)
0.2
0.3
0.4
0.5
0.6
0.7
0
2000
4000
6000
8000
10000
t

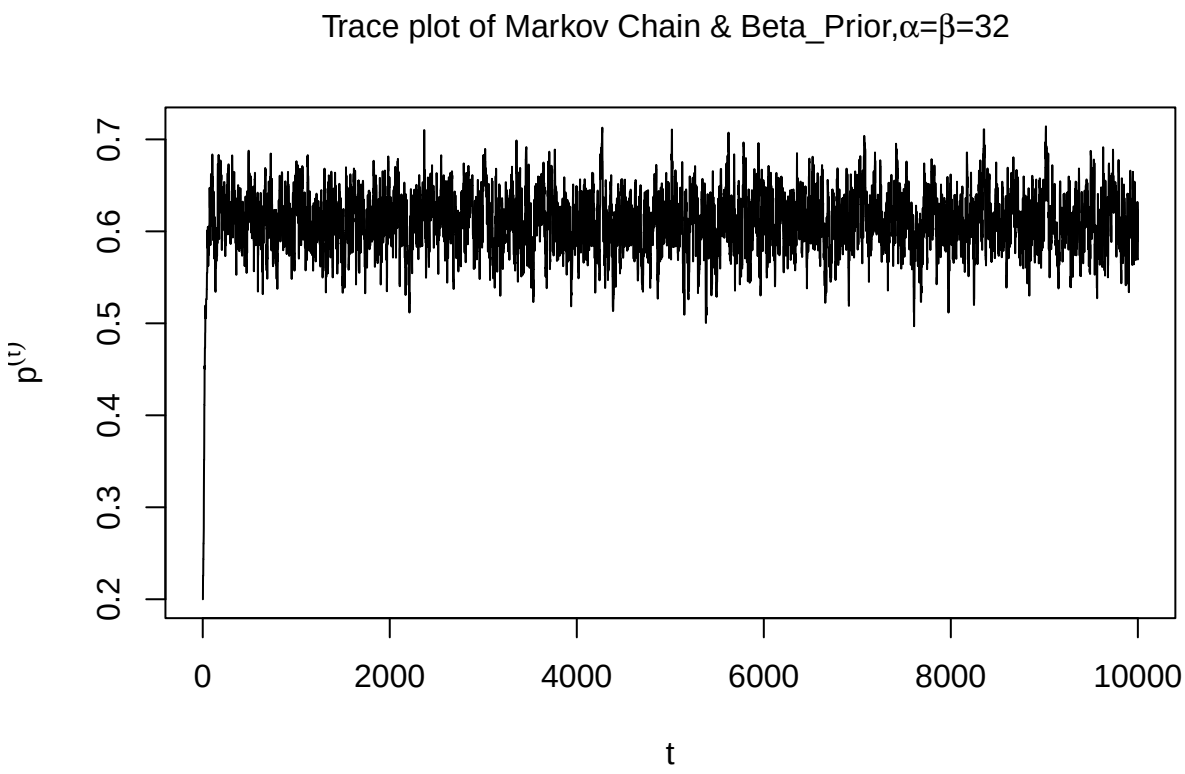
```
## [1] 4
```


Trace plot of Markov Chain & Beta_Prior,α=β=8
p(t)
0.2
0.3
0.4
0.5
0.6
0.7
0
2000
4000
6000
8000
10000
t

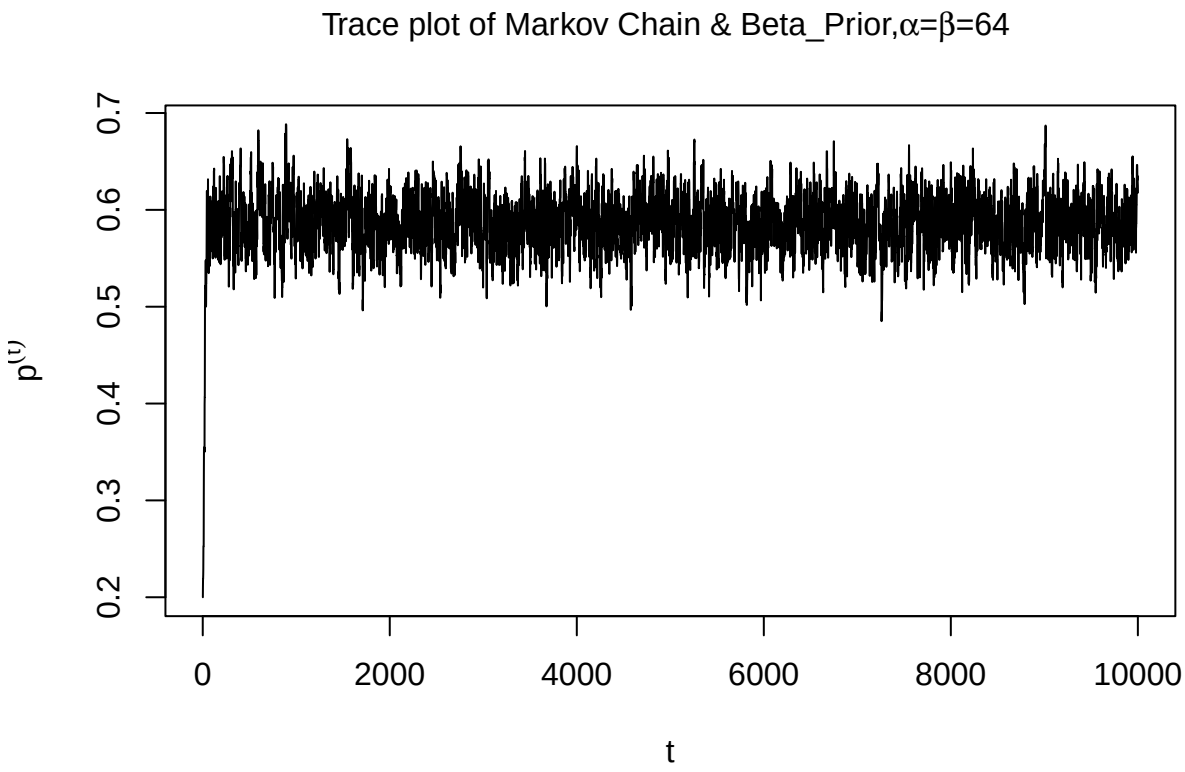
```
## [1] 8
```


Trace plot of Markov Chain & Beta_Prior,α=β=16
p(t)
0.2
0.3
0.4
0.5
0.6
0.7
0
2000
4000
6000
8000
10000
t

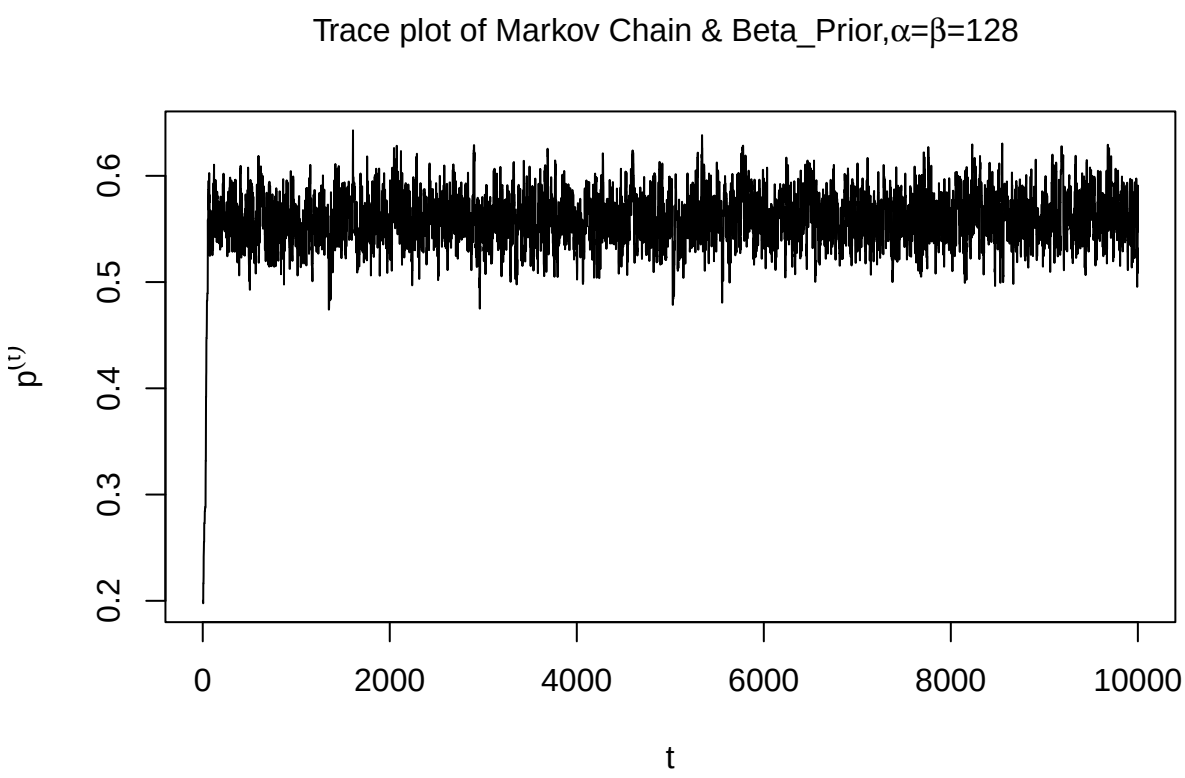
```
## [1] 16
```


Trace plot of Markov Chain & Beta_Prior,α=β=32
p(t)
0.2
0.3
0.4
0.5
0.6
0.7
0
2000
4000
6000
8000
10000
t

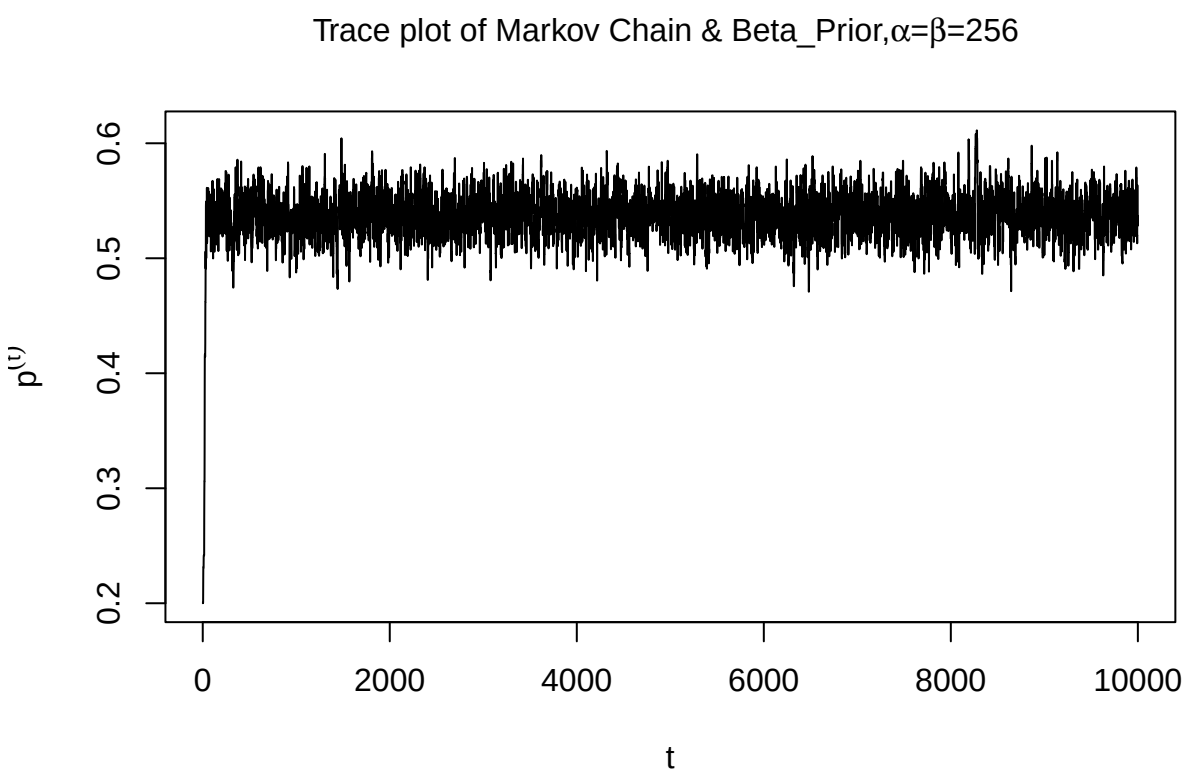
```
## [1] 32
```


Trace plot of Markov Chain & Beta_Prior,α=β=64
p(t)
0.2
0.3
0.4
0.5
0.6
0.7
0
2000
4000
6000
8000
10000
t

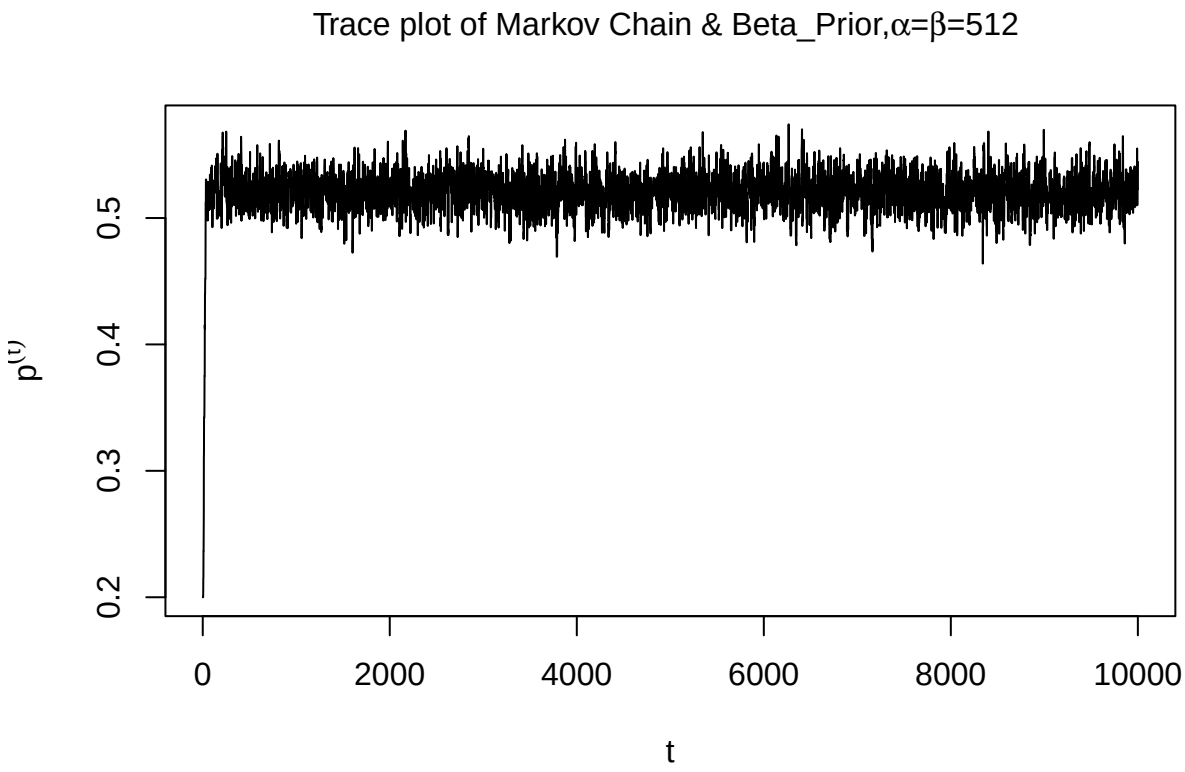
```
## [1] 64
```


Trace plot of Markov Chain & Beta_Prior,α=β=128
p(t)
0.2
0.3
0.4
0.5
0.6
0
2000
4000
6000
8000
10000
t

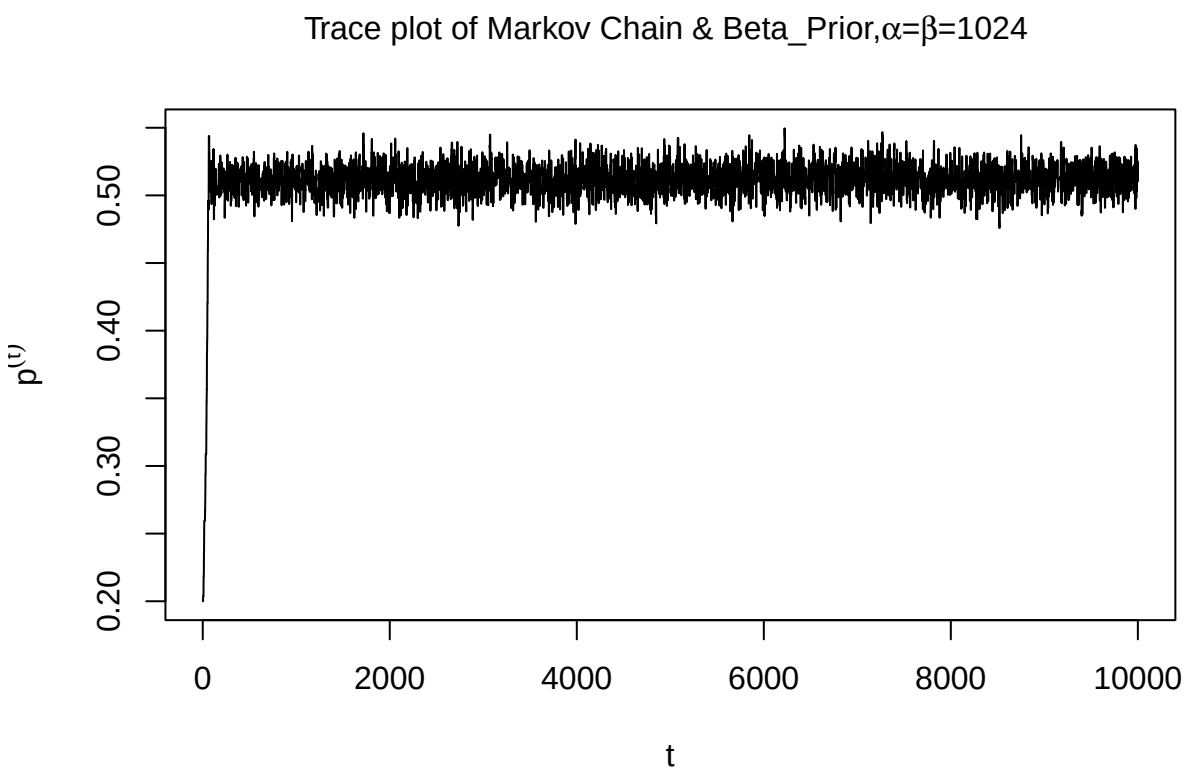
```
## [1] 128
```


Trace plot of Markov Chain & Beta_Prior,α=β=256
p(t)
0.2
0.3
0.4
0.5
0.6
0
2000
4000
6000
8000
10000
t

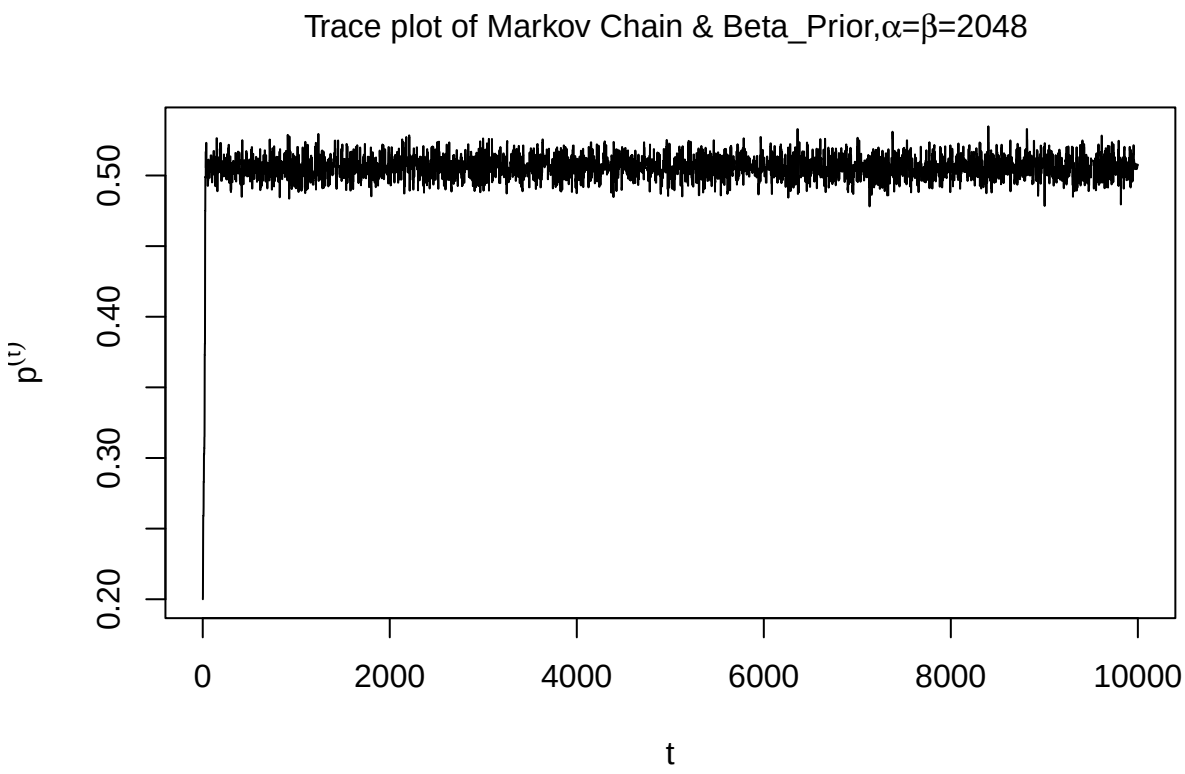
```
## [1] 256
```


Trace plot of Markov Chain & Beta_Prior,α=β=512
p(t)
0.2
0.3
0.4
0.5
0
2000
4000
6000
8000
10000
t

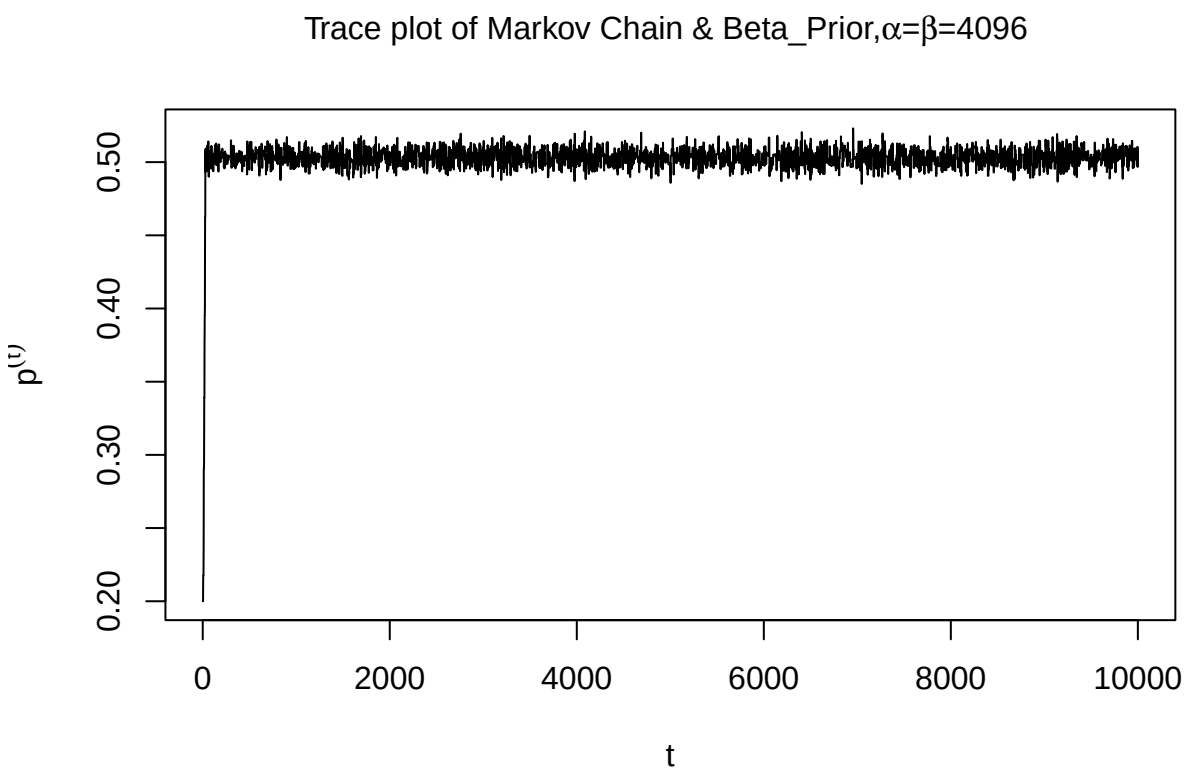
```
## [1] 512
```


Trace plot of Markov Chain & Beta_Prior,α=β=1024
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

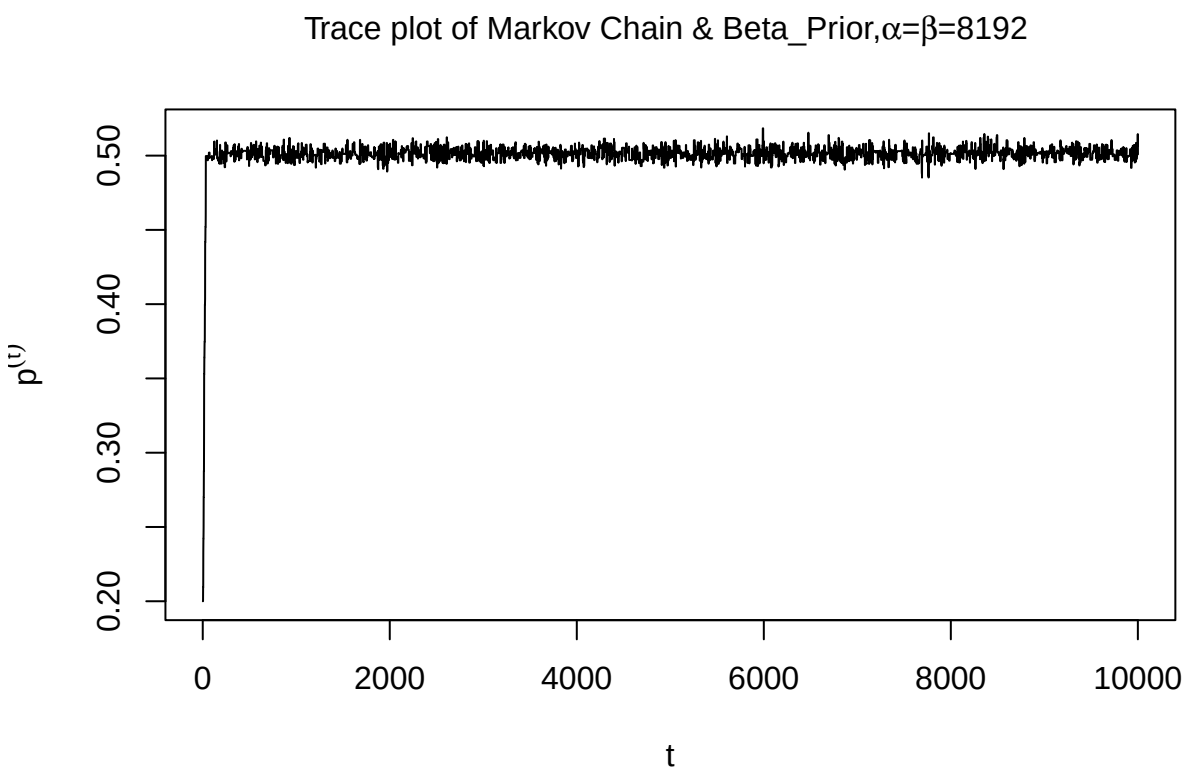
```
## [1] 1024
```


Trace plot of Markov Chain & Beta_Prior,α=β=2048
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

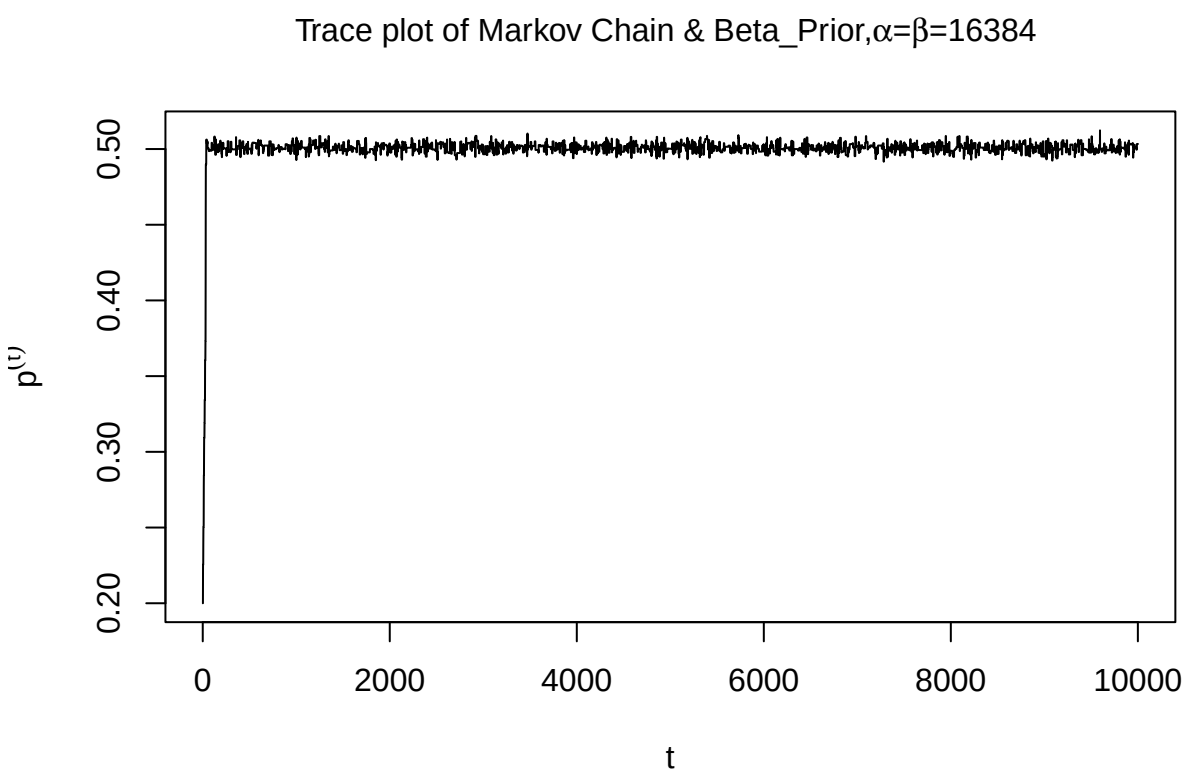
```
## [1] 2048
```


Trace plot of Markov Chain & Beta_Prior,α=β=4096
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

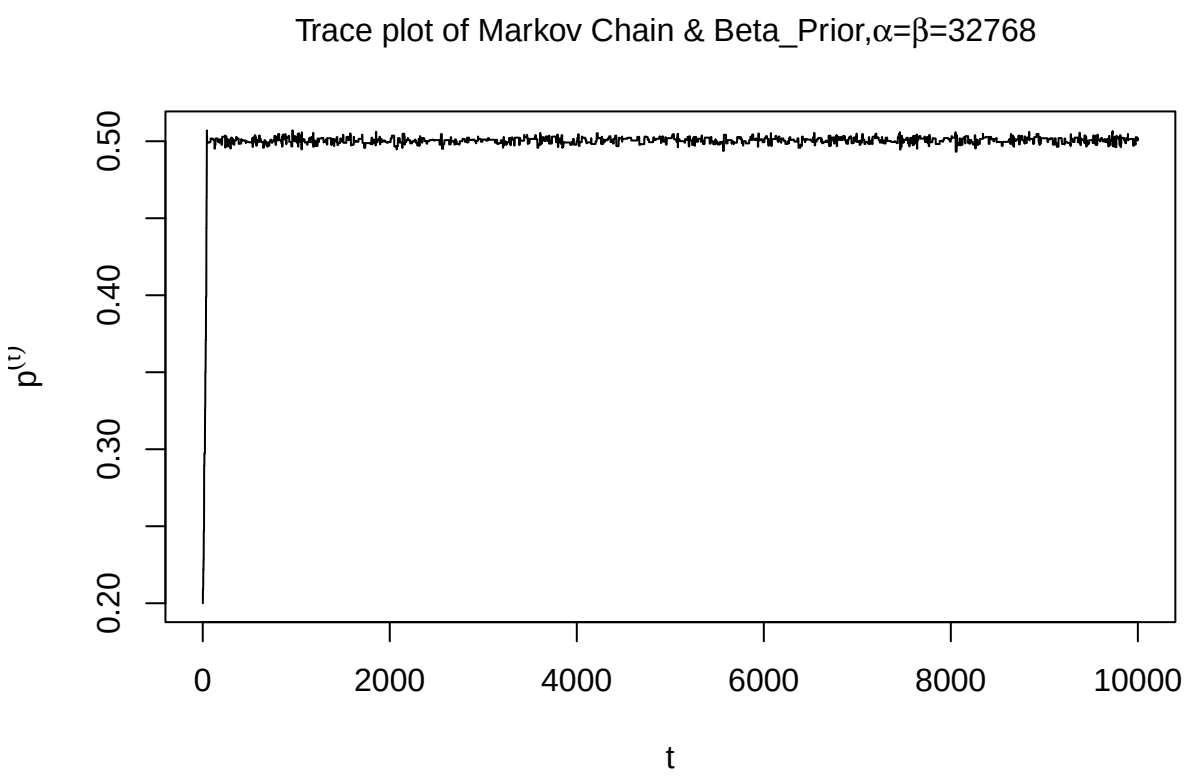
```
## [1] 4096
```


Trace plot of Markov Chain & Beta_Prior,α=β=8192
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

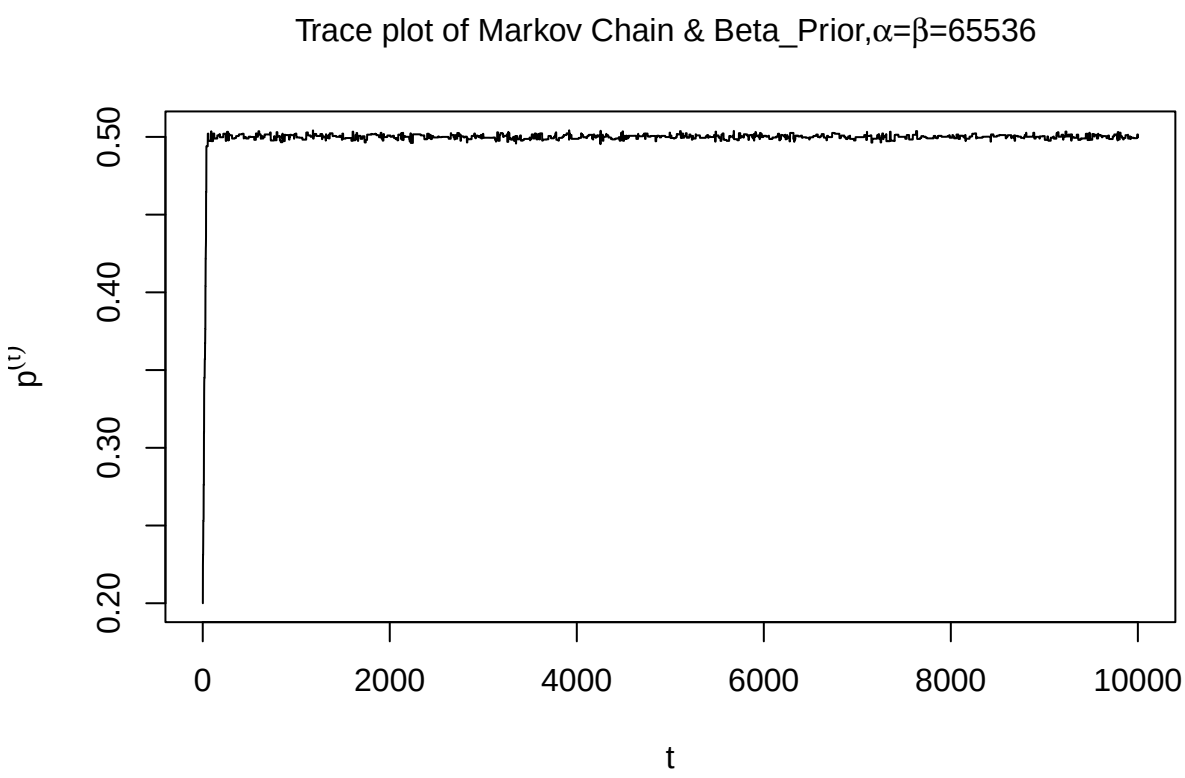
```
## [1] 8192
```


Trace plot of Markov Chain & Beta_Prior,α=β=16384
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

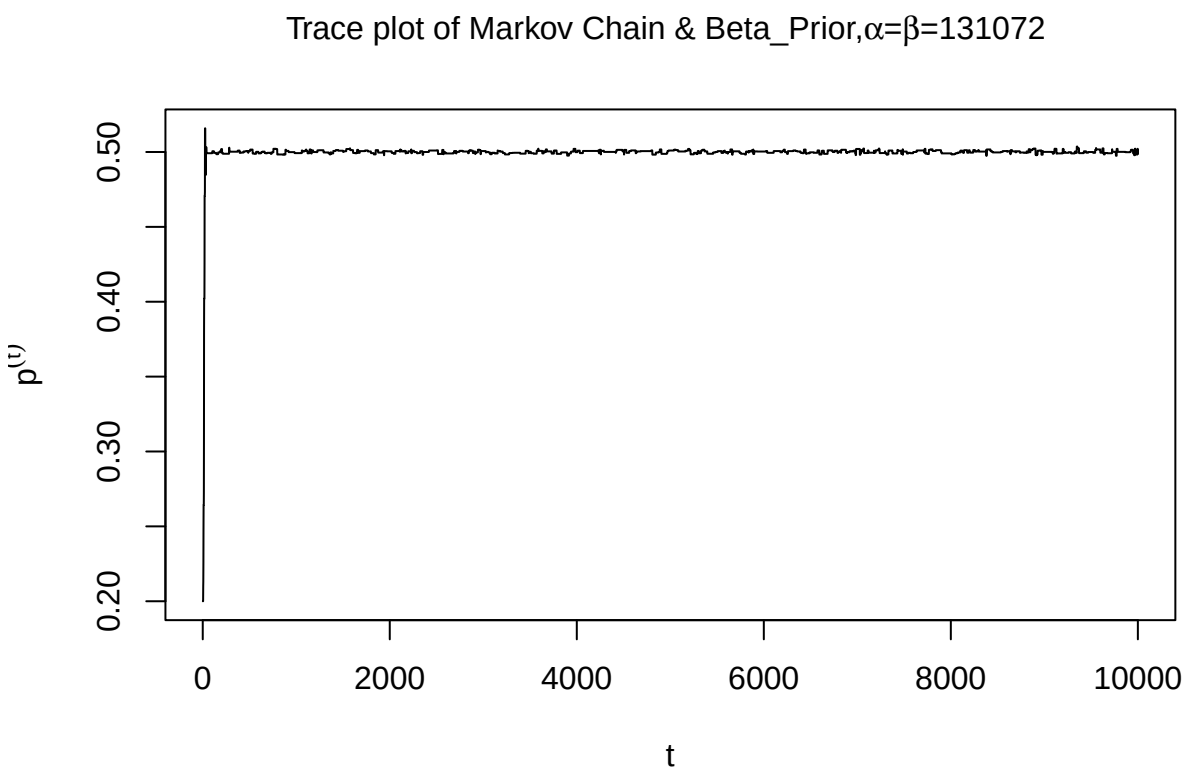
```
## [1] 16384
```


Trace plot of Markov Chain & Beta_Prior,α=β=32768
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

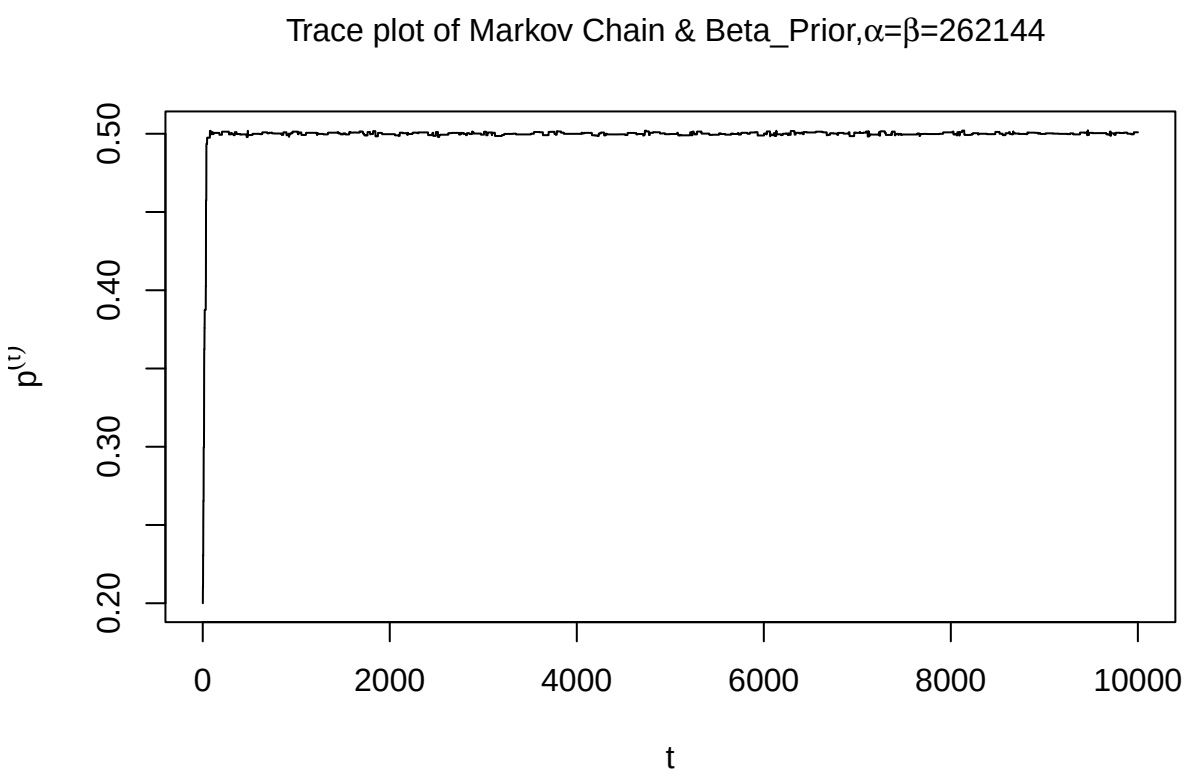
```
## [1] 32768
```


Trace plot of Markov Chain & Beta_Prior,α=β=65536
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

```
## [1] 65536
```

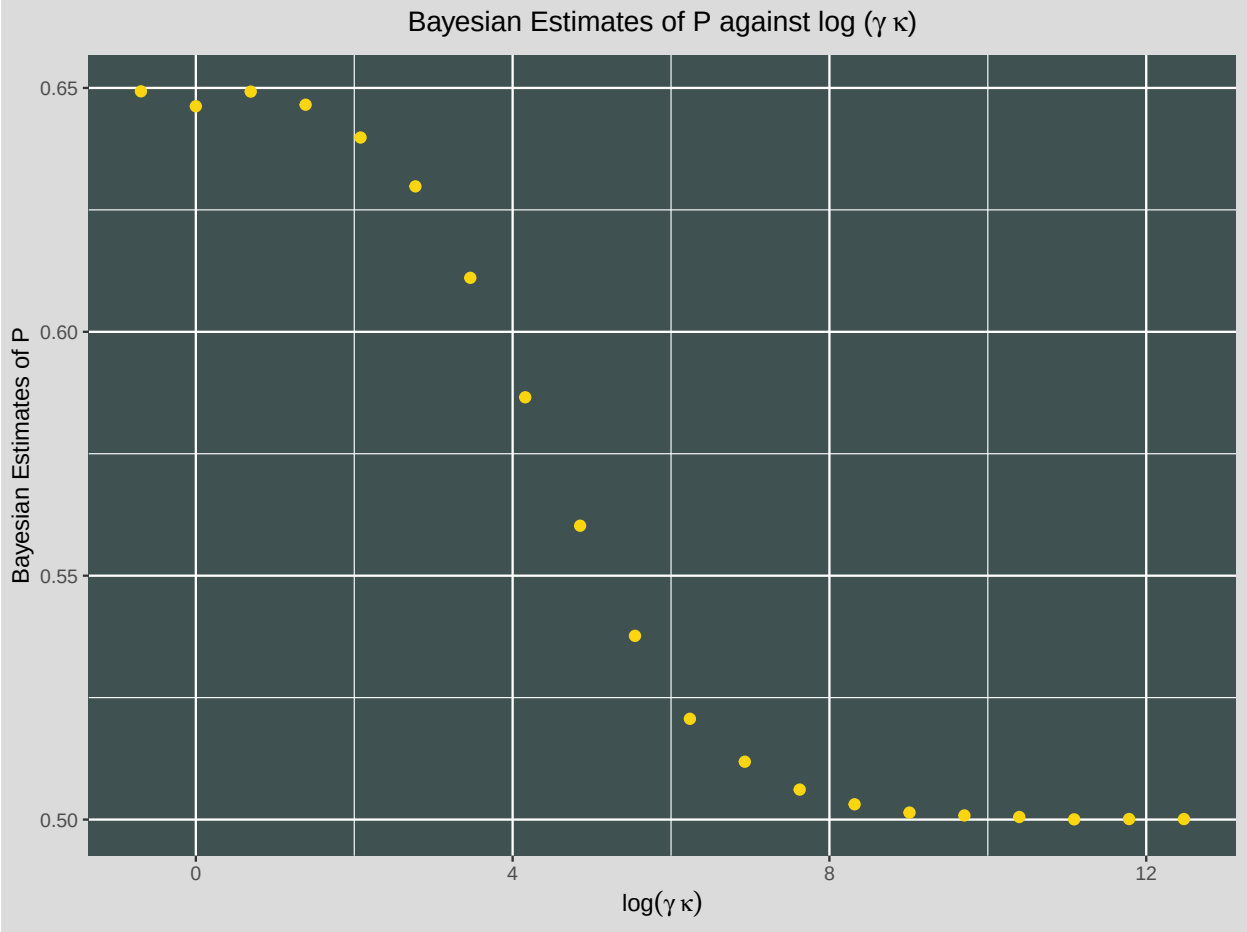

Trace plot of Markov Chain & Beta_Prior,α=β=131072
p(t)
0.20
0.30
0.40
0.50
0
2000
4000
6000
8000
10000
t

```
## [1] 131072
```




```
## [1] 262144
```

```
#p_k
#log_gammak
data_to_plot <- data.frame(estimate_probability = p_k, log_gamma_k = log_gammak)
#data_to_plot
ggplot(data_to_plot) +
     geom_point(aes(x = log_gamma_k, y= estimate_probability), color = "#FAD510", size =2) + ggtitle(bqu
  labs(x = bquote(log(gamma ~ kappa)), y = "Bayesian Estimates of P") +
  theme(plot.title = element_text(hjust = 0.5), panel.background = element_rect(fill = "#3F5151"),plot.
```

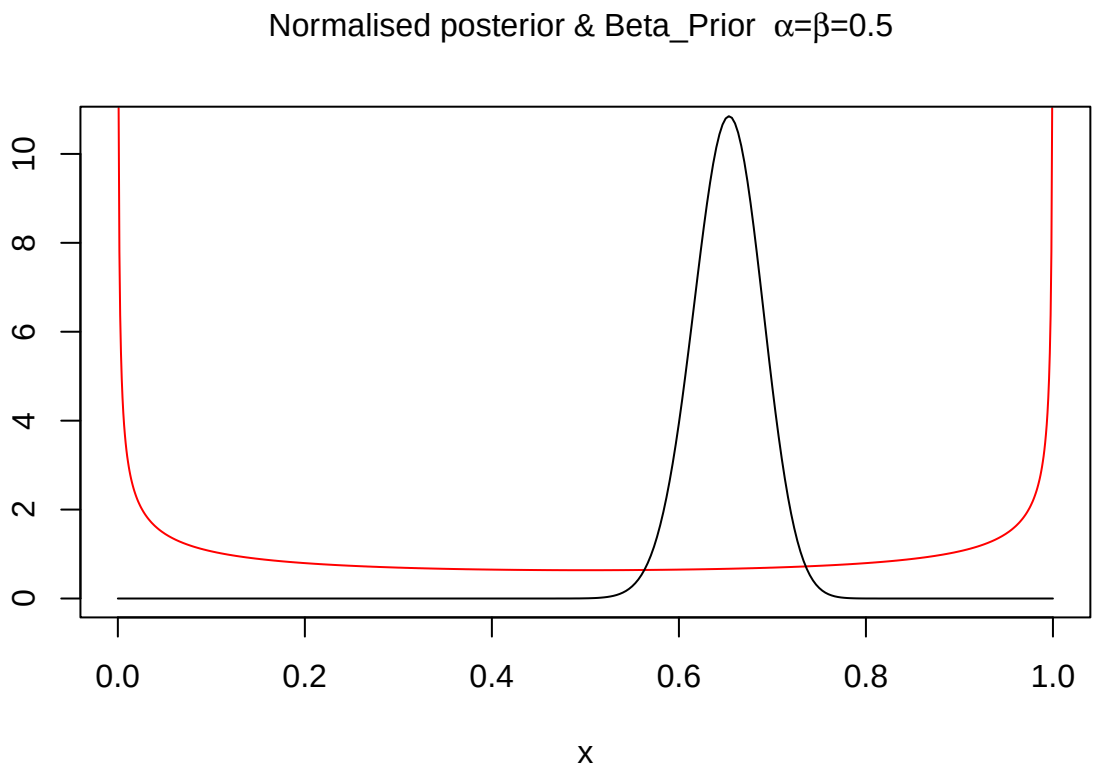


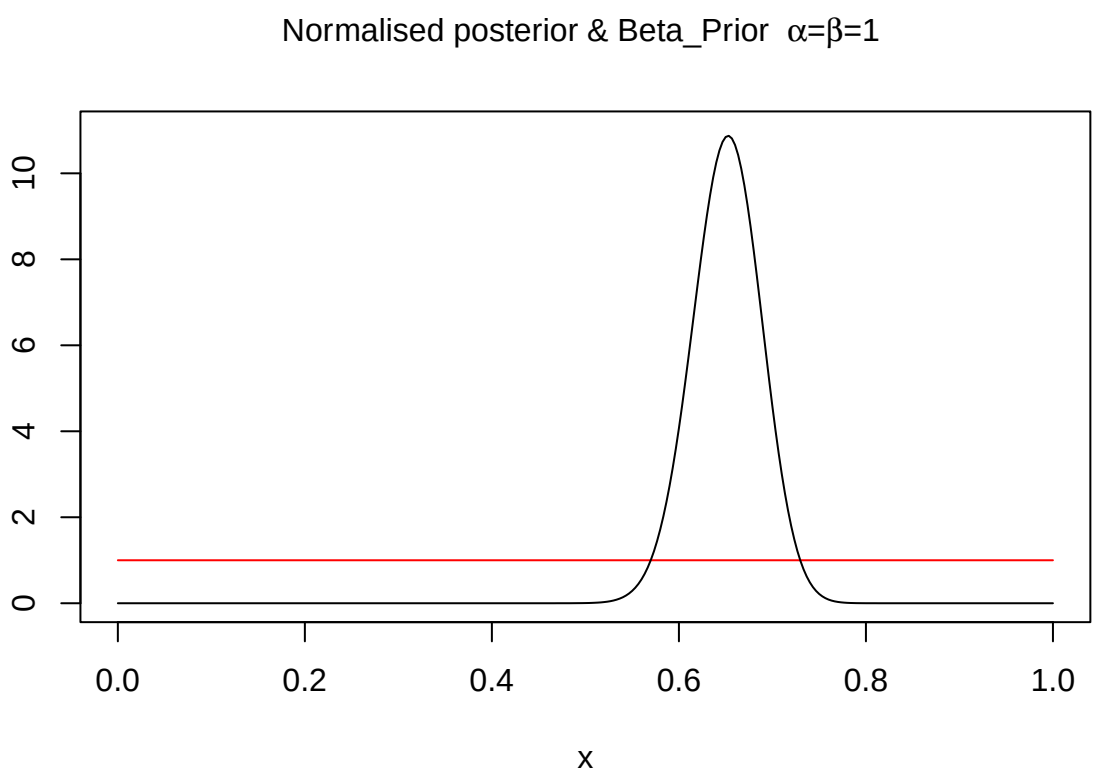

```
## Below calculate the intergrand in order to plot posteria and prior distribution in the same scale sa

for ( ii in 1:length(k_seq)) {

alpha <-2^k_seq[ii]
beta <- 2^k_seq[ii]
p = seq(0,1, length=100)
integrand<-function(x)x^(108+alpha)*(1-x)^(57+beta)
print(integrate(integrand, lower= 0,upper=1)$value)
}
```

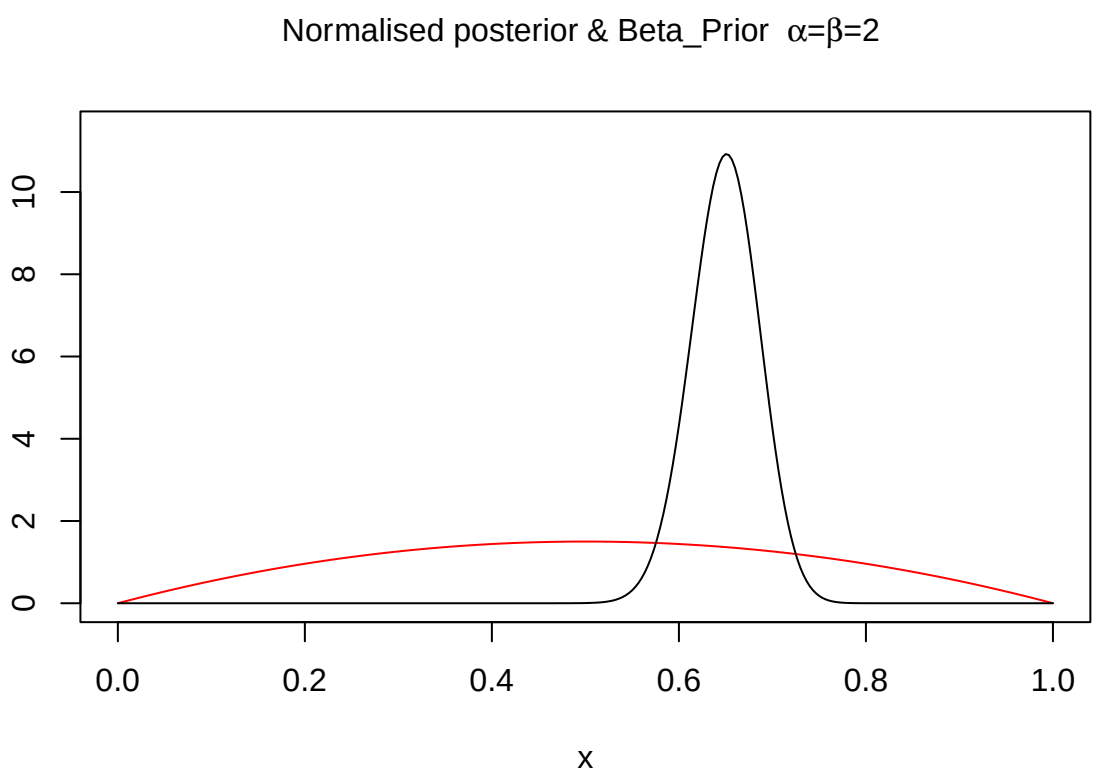
```
## [1] 2.829425e-48
## [1] 1.343539e-48
## [1] 3.035004e-49
## [1] 1.559773e-50
## [1] 4.228698e-53
## [1] 3.397652e-58
## [1] 2.864316e-68
## [1] 3.913474e-88
## [1] 2.472776e-127
## [1] 5.233888e-205
## [1] 0
## [1] 0
```

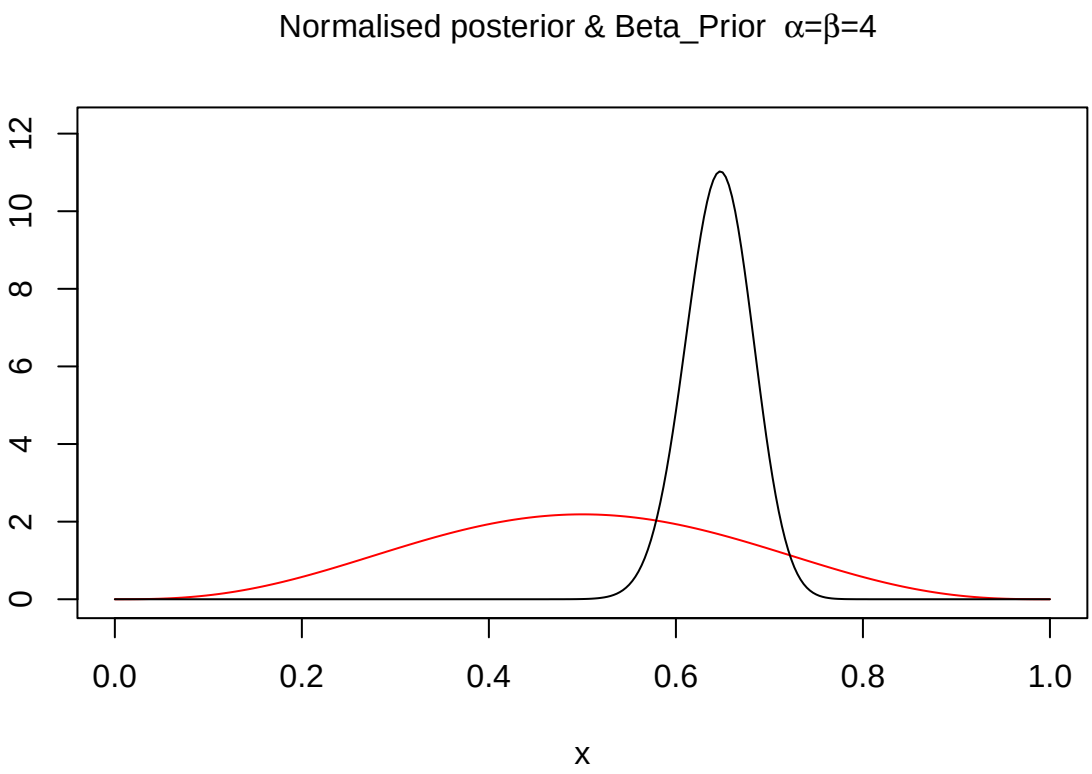
```
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
```

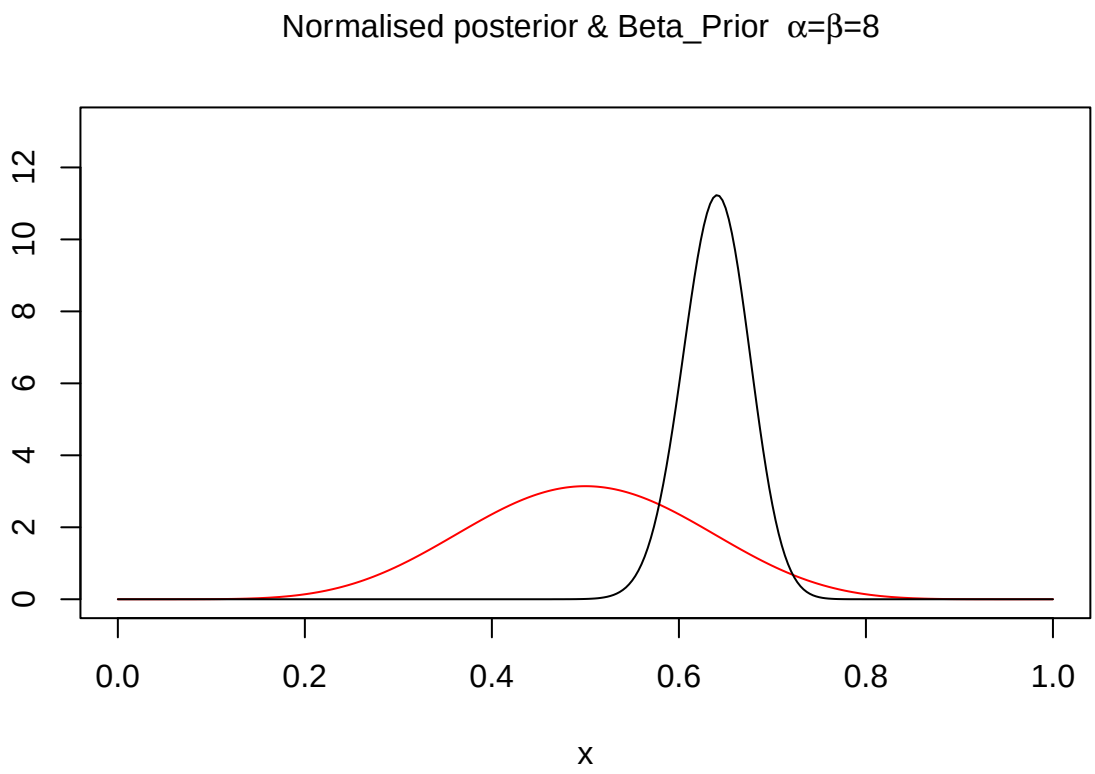
```r
for ( ii in 1:length(k_seq)) {

alpha <-2^k_seq[ii]
beta <- 2^k_seq[ii]
p = seq(0,1, length=100)
#plot(p, dbeta(p, alpha, beta), type='l')
integrand<-function(x)x^(108+alpha)*(1-x)^(57+beta)
norm.const<-integrate(integrand, lower= 0,upper=1)$value
curve(dbeta(x, alpha, beta),from=0,to =1,n=1201,col="red",ylab="", ylim = c(0,dbeta(0.5, alpha, beta)+1
ifelse(norm.const==0, curve(x^(108+alpha)*(1-x)^(57+beta), add = TRUE, from=0, to=1, n=601, ylab="", xla


}
```

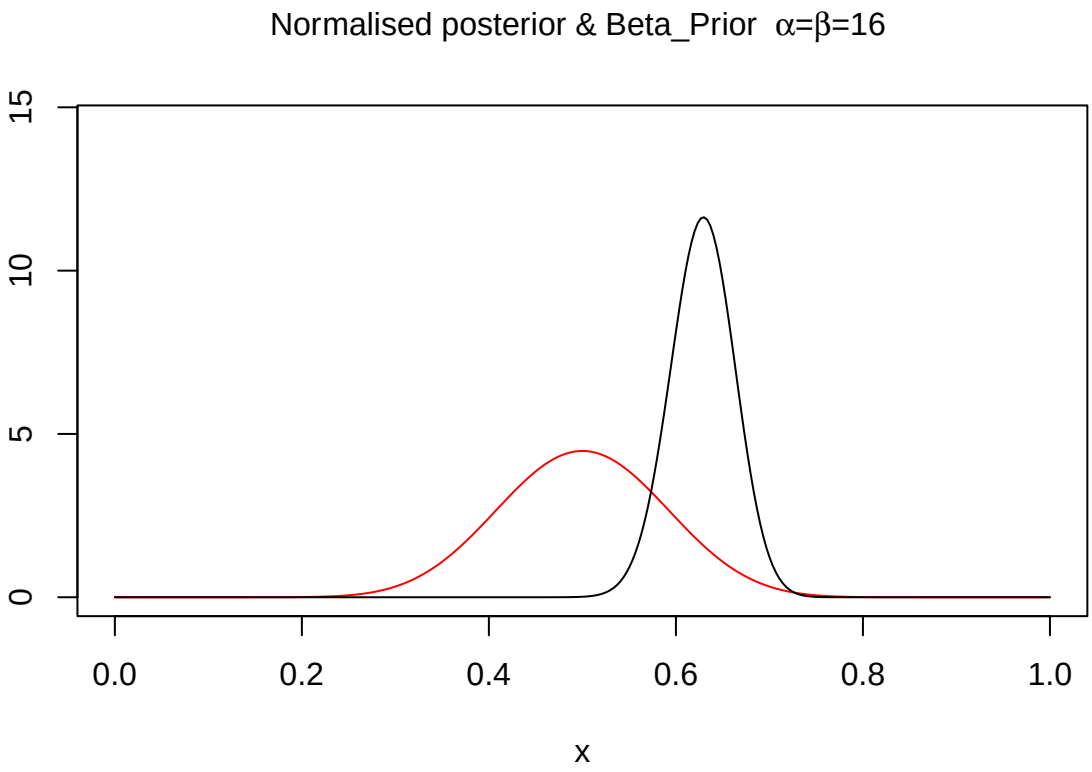


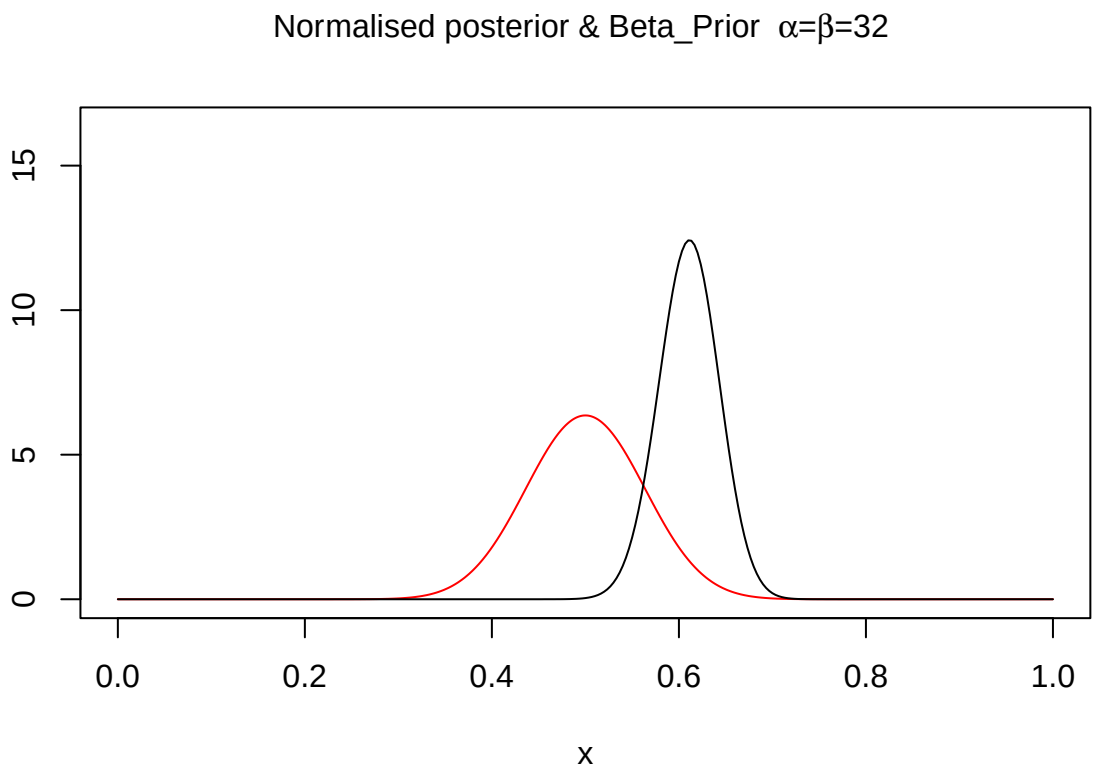

Normalised posterior & Beta_Prior α=β=1
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0
x

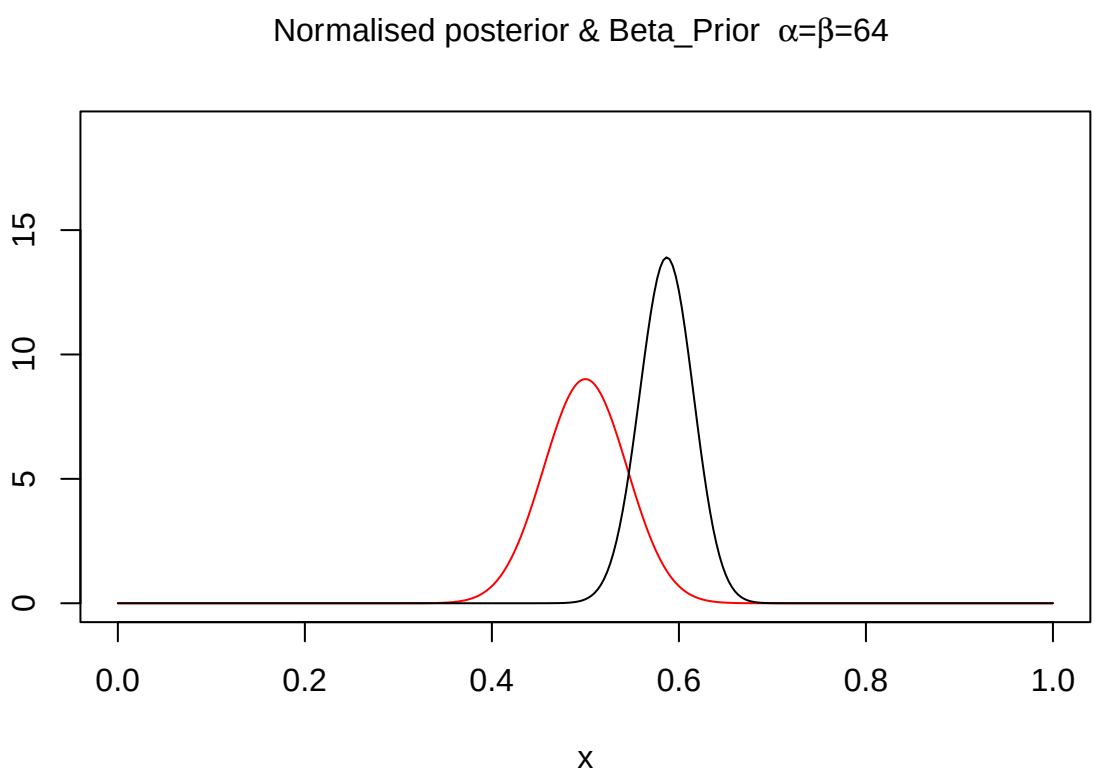

Normalised posterior & Beta_Prior α=β=2
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0
x

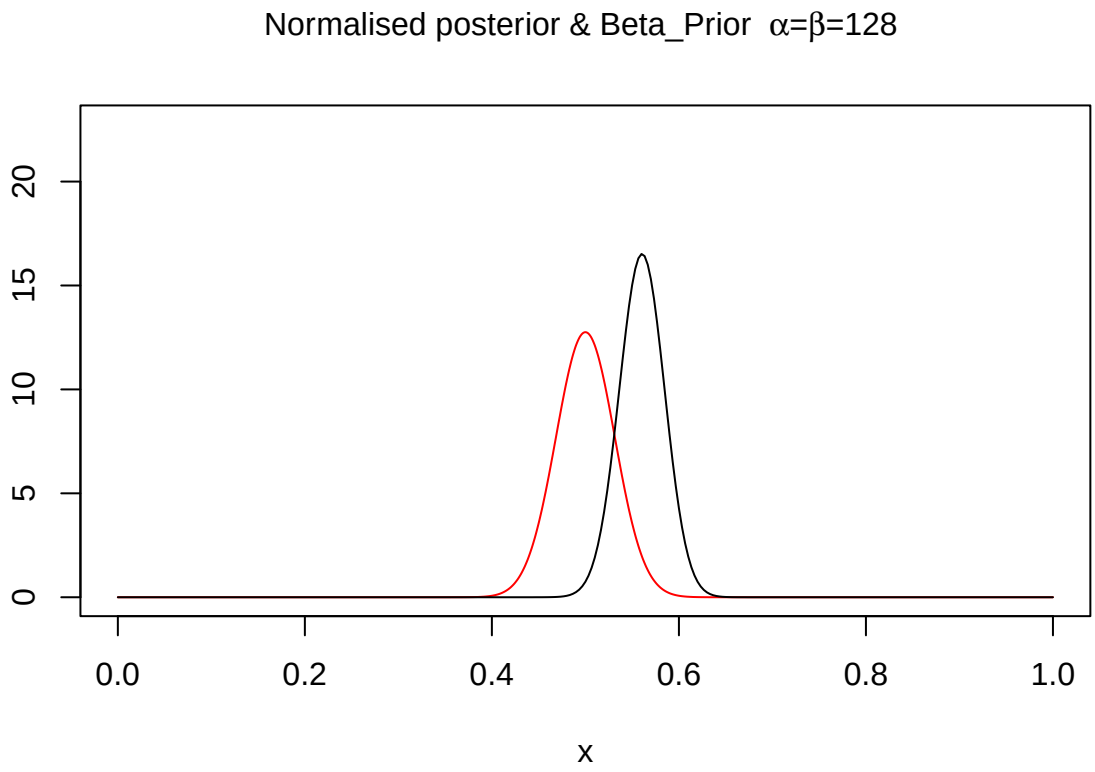

Normalised posterior & Beta_Prior α=β=4
0
2
4
6
8
10
12
0.0
0.2
0.4
0.6
0.8
1.0
x

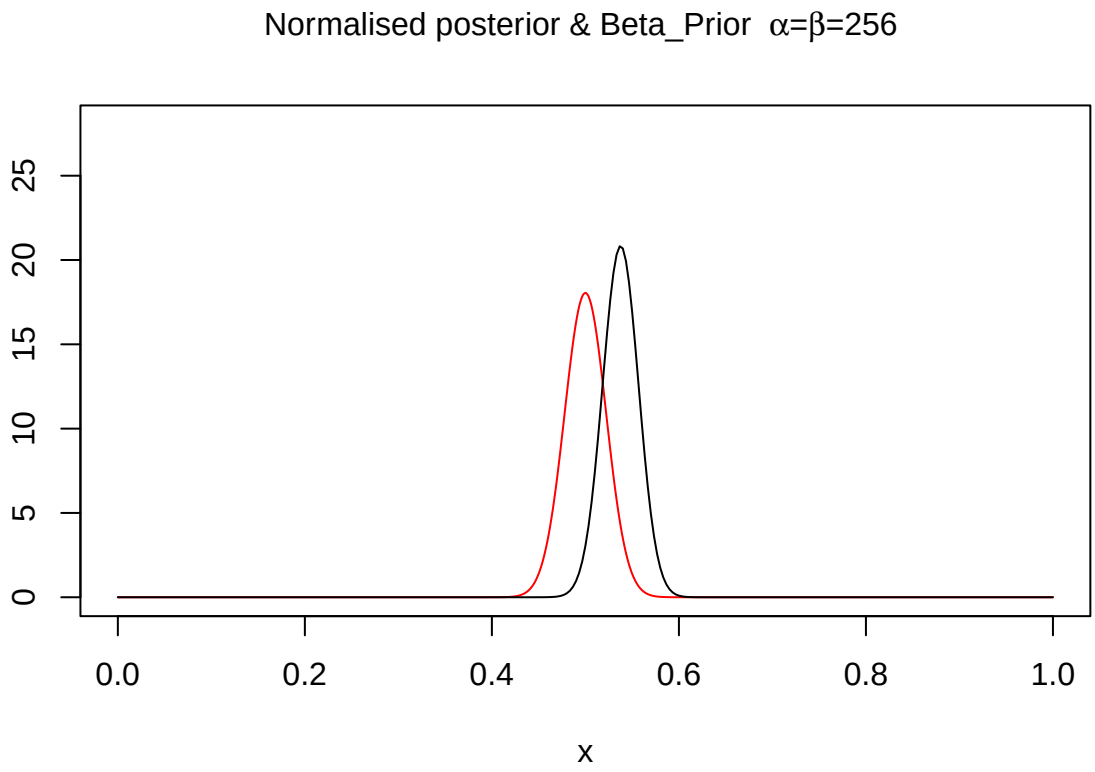

Normalised posterior & Beta_Prior α=β=8
0
2
4
6
8
10
12
0.0
0.2
0.4
0.6
0.8
1.0
x

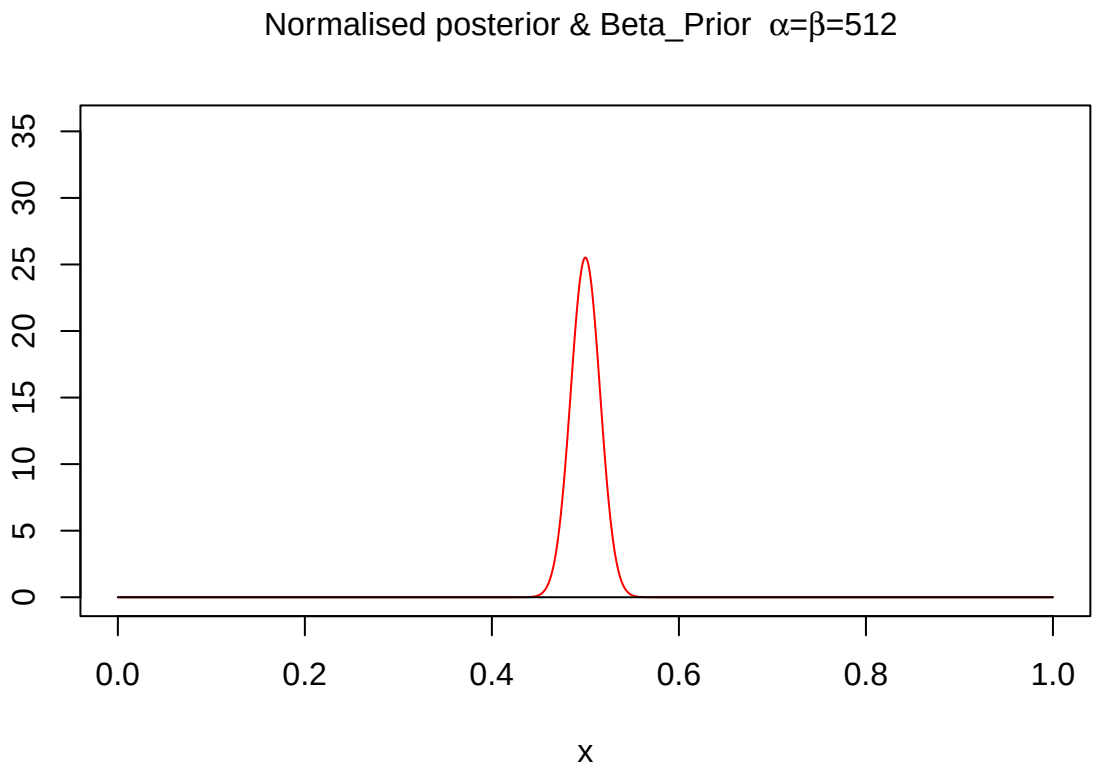

Normalised posterior & Beta_Prior α=β=16
0
5
10
15
0.0
0.2
0.4
0.6
0.8
1.0
x

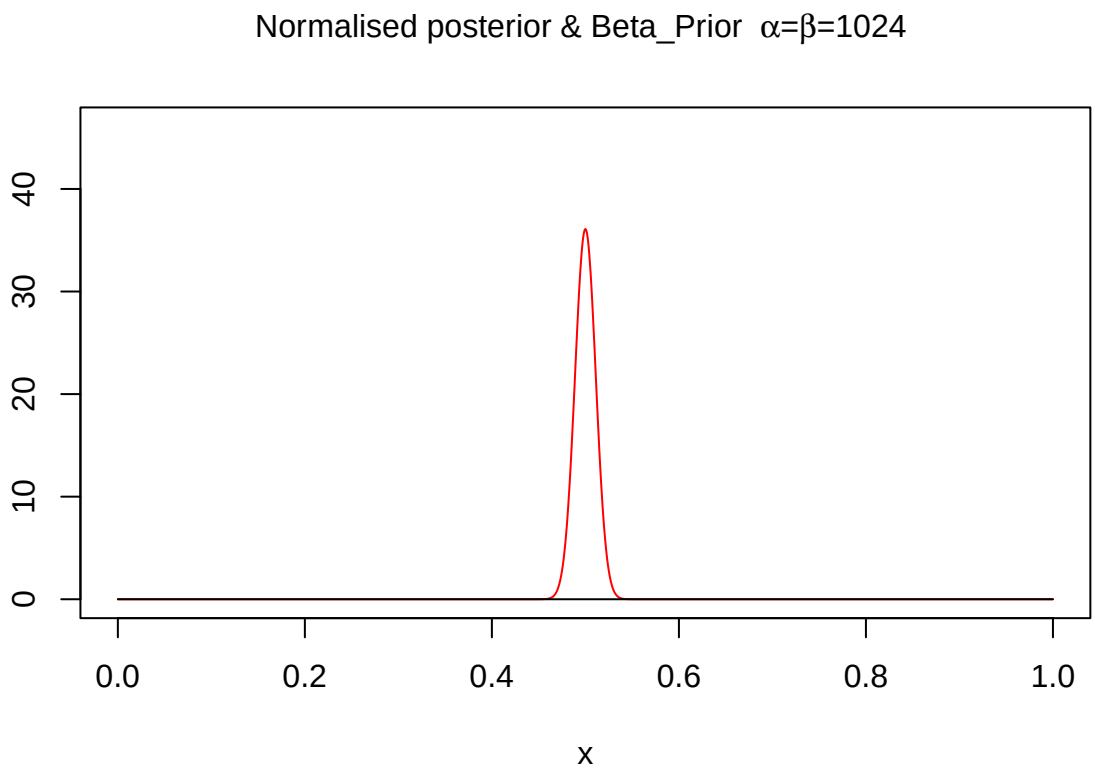

Normalised posterior & Beta_Prior α=β=32
0
5
10
15
0.0
0.2
0.4
0.6
0.8
1.0
x

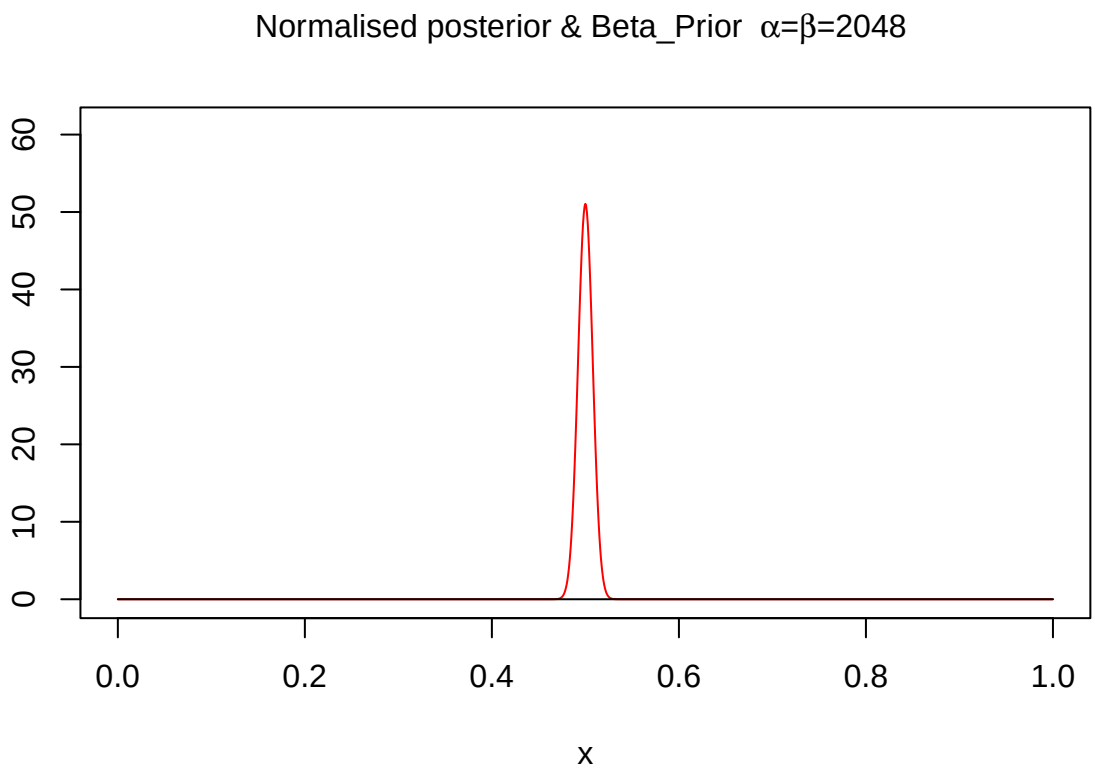

Normalised posterior & Beta_Prior α=β=64
0
5
10
15
0.0
0.2
0.4
0.6
0.8
1.0
x

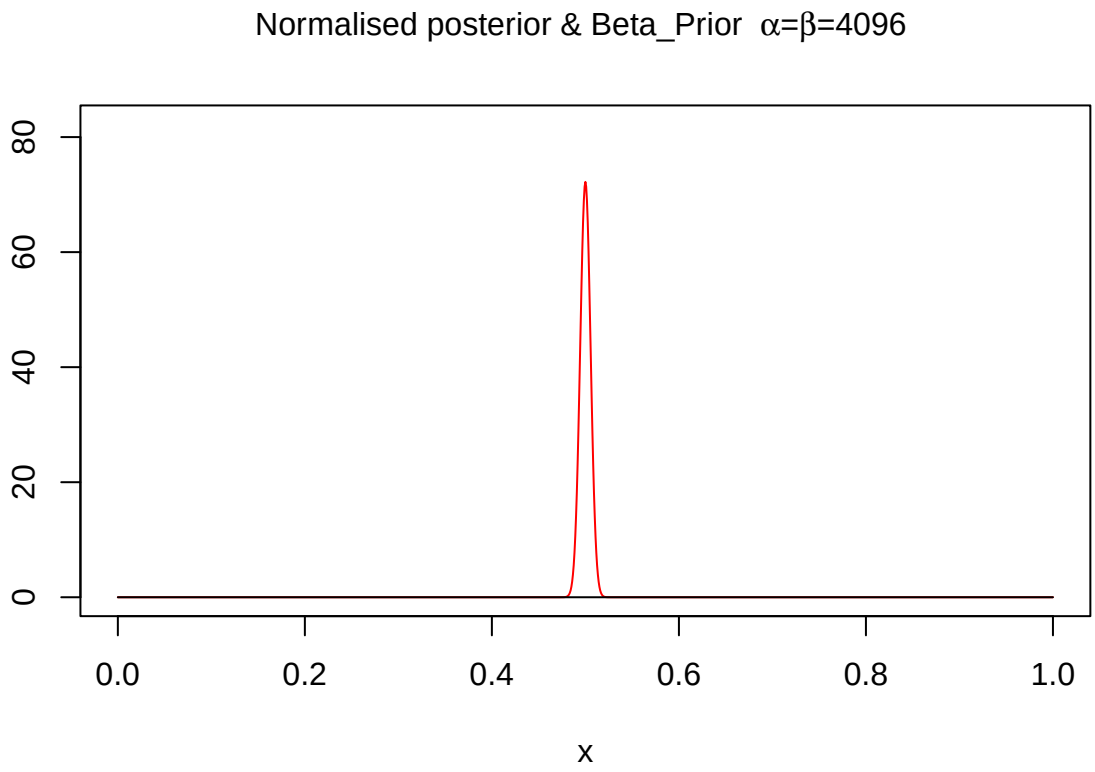

Normalised posterior & Beta_Prior α=β=128
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0
x

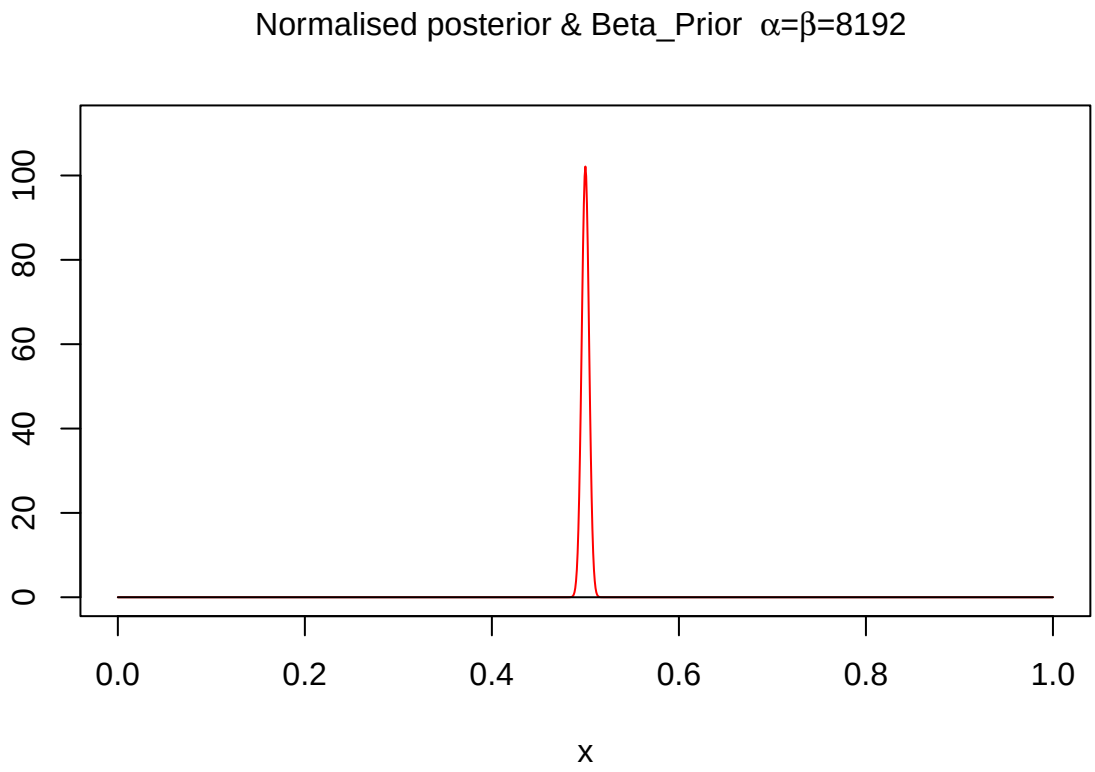

Normalised posterior & Beta_Prior α=β=256
0
5
10
15
20
25
0.0
0.2
0.4
0.6
0.8
1.0
x

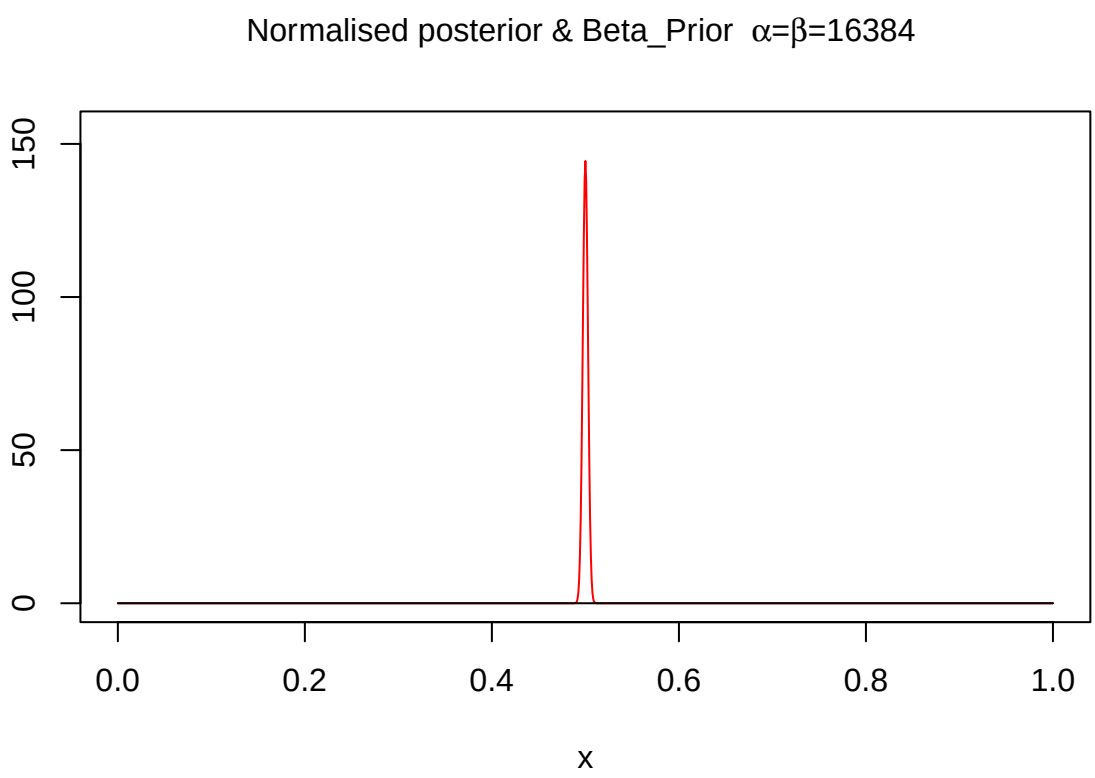

Normalised posterior & Beta_Prior α=β=512
0
5
10
15
20
25
30
35
0.0
0.2
0.4
0.6
0.8
1.0
x

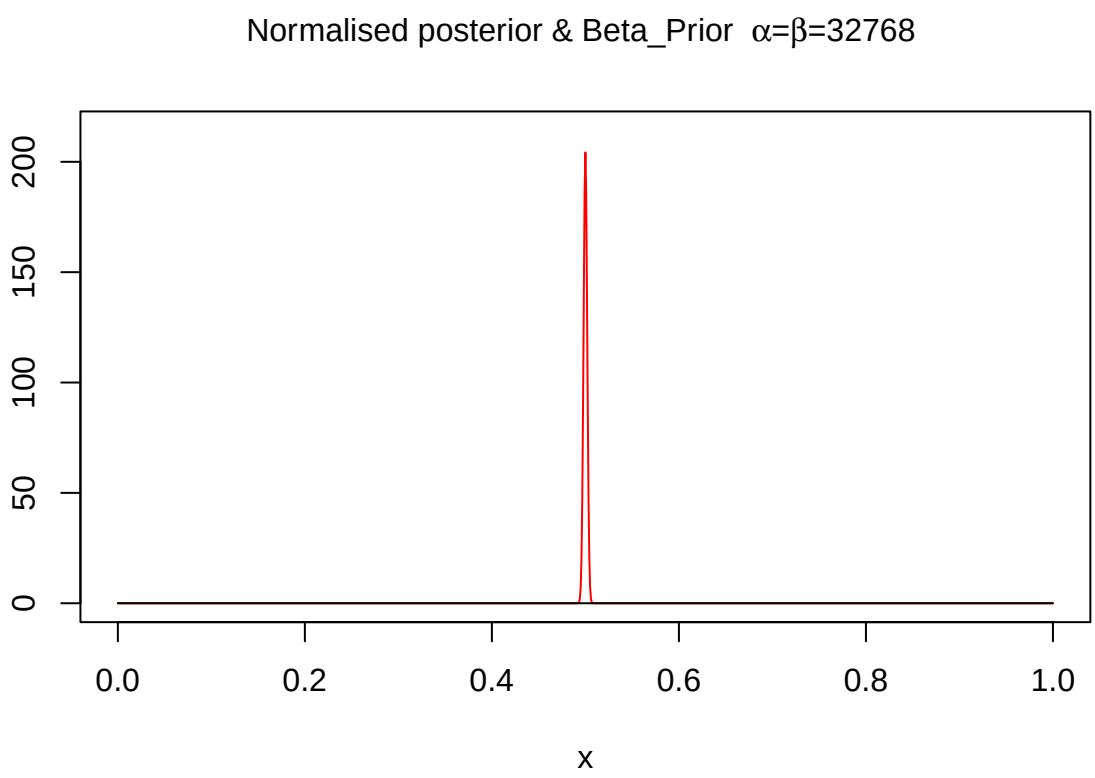

Normalised posterior & Beta_Prior α=β=1024
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0
x

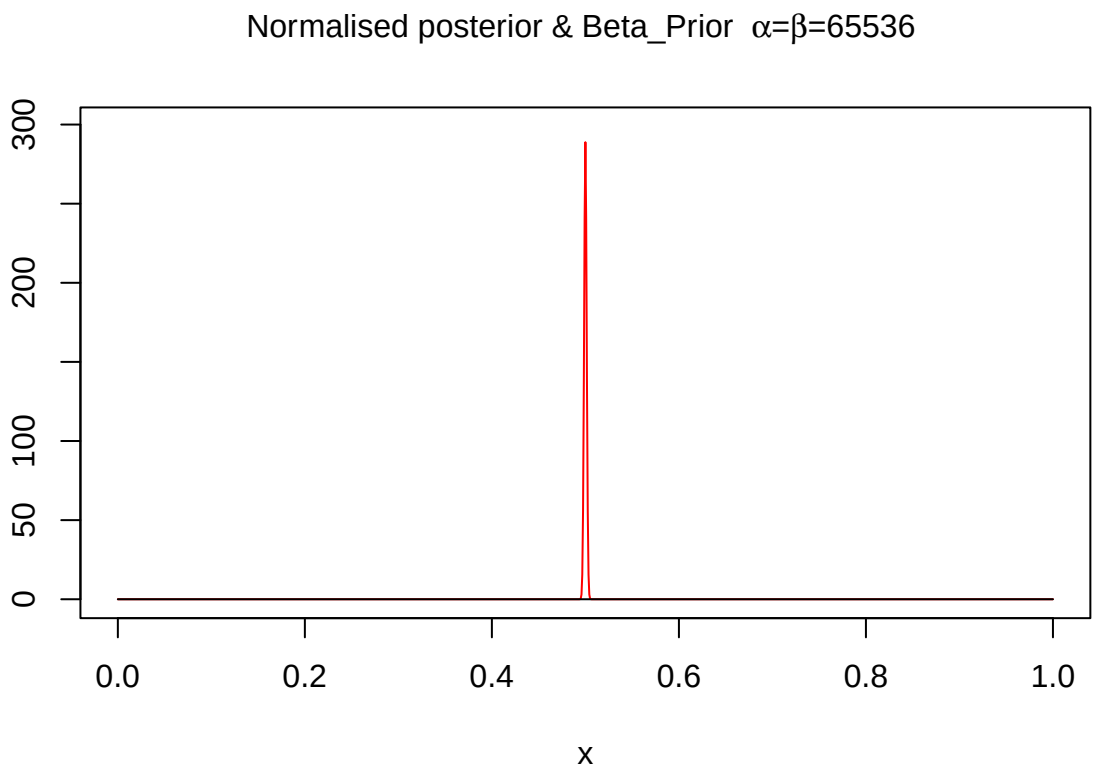

Normalised posterior & Beta_Prior α=β=2048
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0
x

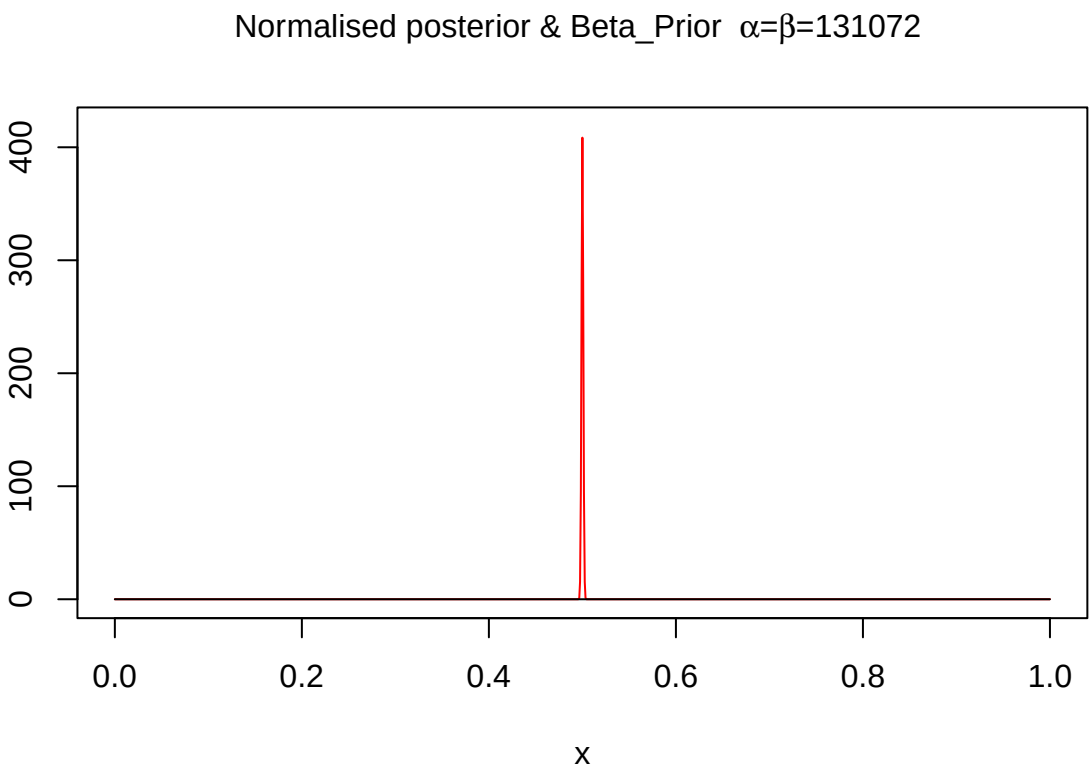

Normalised posterior & Beta_Prior α=β=4096
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
x

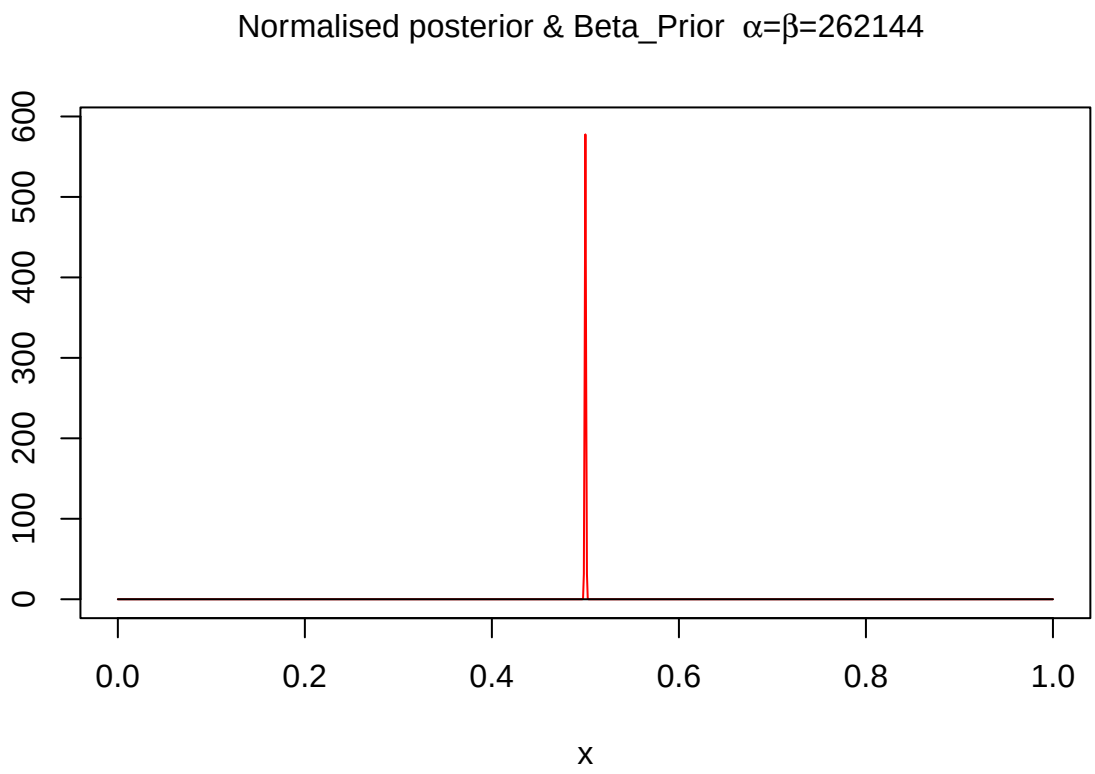

Normalised posterior & Beta_Prior α=β=8192
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
x


Normalised posterior & Beta_Prior α=β=16384
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
x


Normalised posterior & Beta_Prior α=β=32768
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0
x


Normalised posterior & Beta_Prior α=β=65536
300
200
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
x


Normalised posterior & Beta_Prior α=β=131072
0
100
200
300
400
0.0
0.2
0.4
0.6
0.8
1.0
x



TASK 3

```
y <- c(16,9,10,13,19,20,18,17,35,55)
m <- c(74,99,58,70,122,77,104,129,308,119)
N <- length(y)
#dbinom(yi, size = mi, prob = 0.4)

data.in <- list(N=N, y=y, m=m)
data.in
```

```
## $N
## [1] 10
##
## $y
##  [1] 16  9 10 13 19 20 18 17 35 55
##
## $m
##  [1]  74  99  58  70 122  77 104 129 308 119
```

```
data {
  int <lower=1> N;
  int <lower=0> y[N];
  int <lower=1> m[N];
}

parameters {
```

```
  real <lower=0, upper=1> p;  // probability of success (binomial parameter)
}

model {
  // likelihood
  for (i in 1:N) {
    y[i] ~ binomial(m[i], p);  // binomial likelihood
  }

  // prior
  p ~ beta(1, 1);
}
```

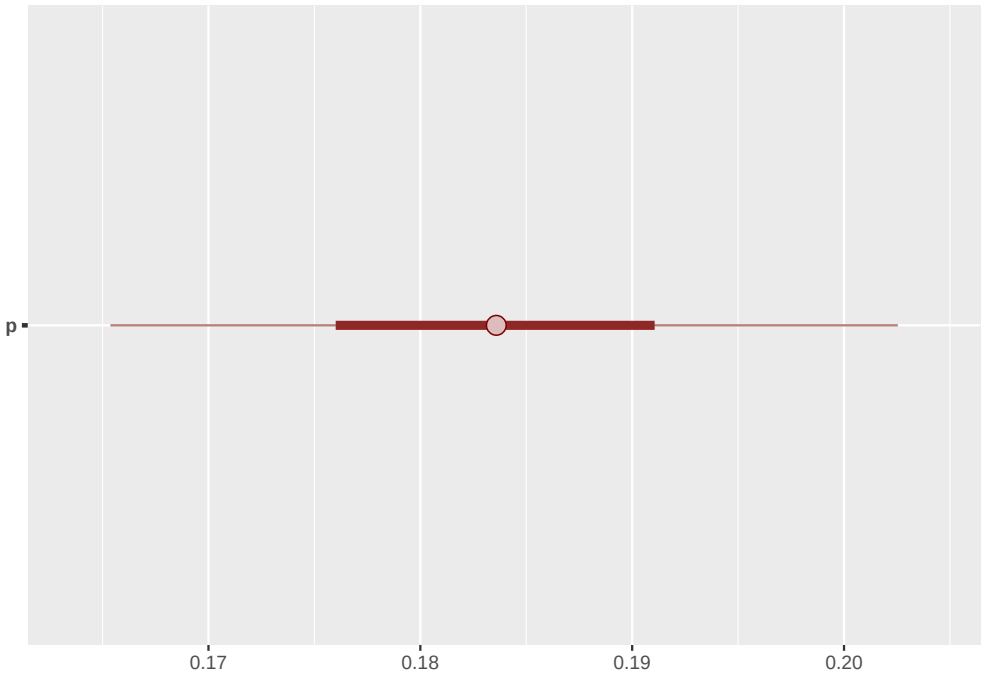
```
model.fit1 <- sampling(Bin_beta_p, data=data.in)
print(model.fit1, pars="p", probs=c(0.1,0.5,0.9),digits=5)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##      mean se_mean      sd     10%     50%     90% n_eff    Rhat
## p 0.18359 0.00028 0.01137 0.16921 0.18331 0.19871  1603 1.00208
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 12 21:40:06 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

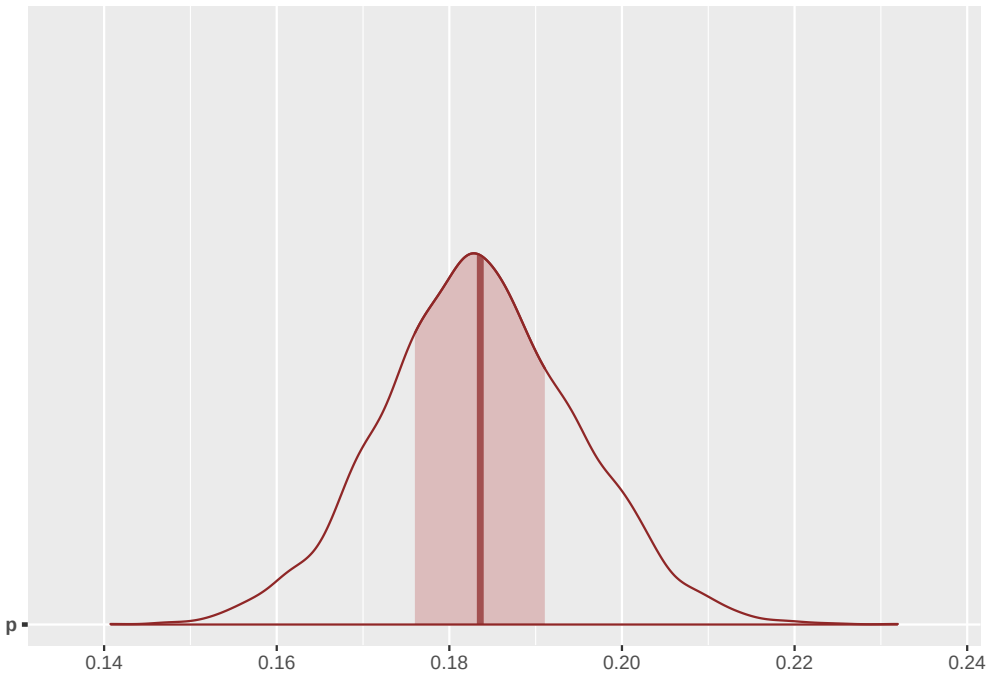
```
check_hmc_diagnostics(model.fit1)
```

```
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

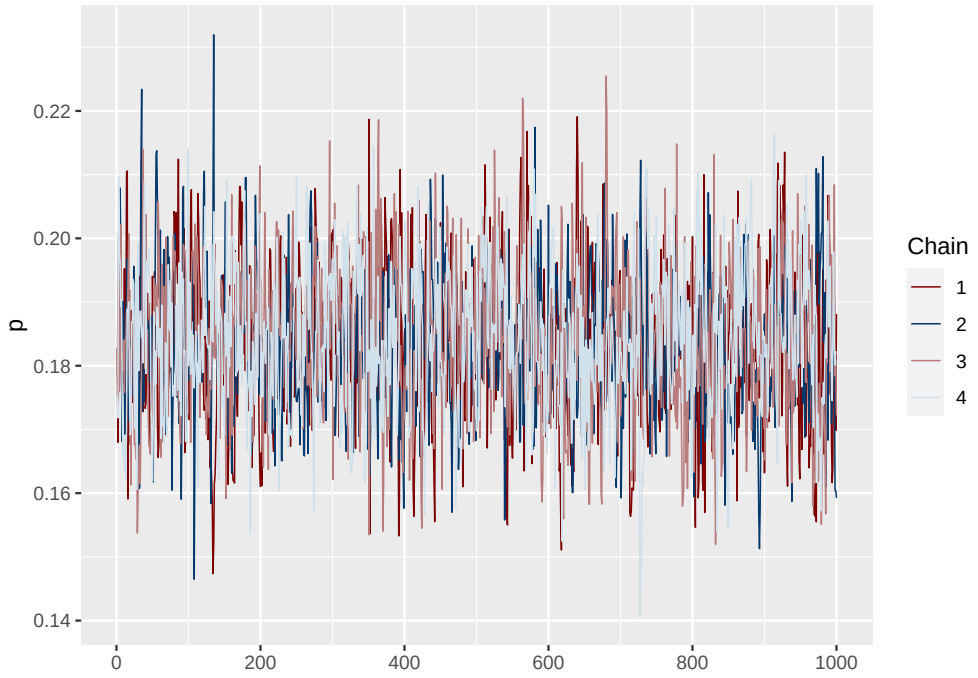
```
posterior <- as.array(model.fit1)
color_scheme_set("red")
mcmc_intervals(posterior, pars="p", point_est = "mean")
```



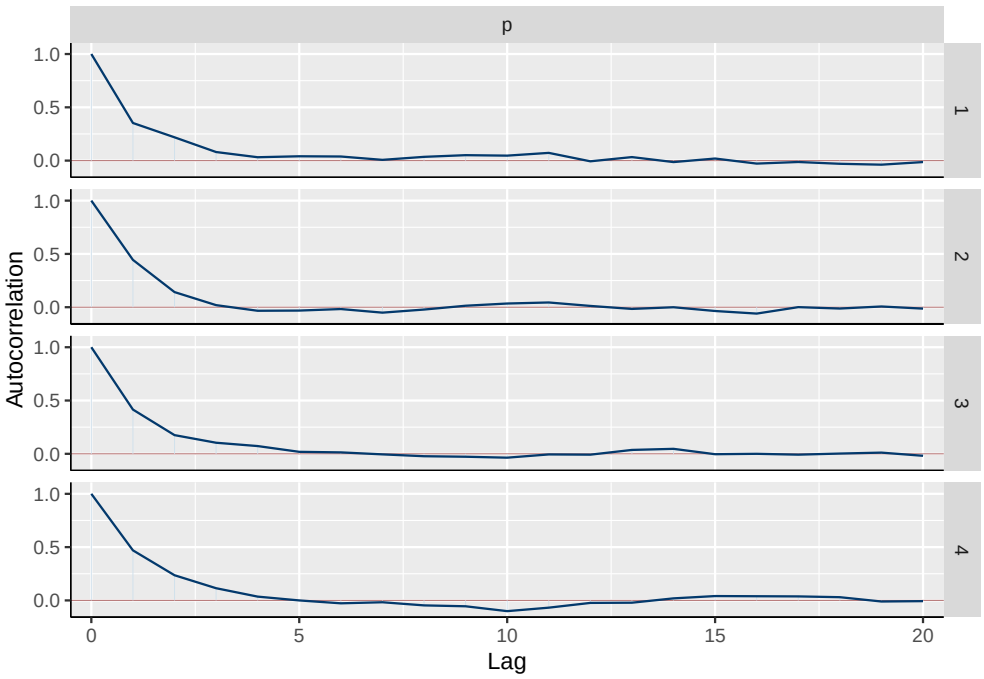

```
mcmc_areas(posterior, pars="p", point_est="mean")
```




```
color_scheme_set("mix-blue-red")
mcmc_trace(posterior, pars="p")
```




```
mcmc_acf(posterior, pars="p")
```




```
data {
  int <lower=1> N;          // number of observations
  int <lower=0> y[N];       // number of successes
  int <lower=1> m[N];       // number of trials
}
```

```
parameters {
  real <lower=0, upper=1> pis[N];  // probability of success (binomial parameter)
}

model {
  // likelihood
  for (i in 1:N) {
    y[i] ~ binomial(m[i], pis[i]);  // binomial likelihood
  }

  // prior
  pis ~ beta(1, 1);  // uniform prior over (0, 1)
}

generated quantities {
  real r;
  r = min(pis) / max(pis);
  real r_mean;
  r_mean = mean(pis);

}
```

```
model.fit2 <- sampling(Bin_beta_pi, data=data.in , iter = 4000)
print(model.fit2, pars=c("pis","r","r_mean"),digits=5)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=4000; warmup=2000; thin=1;
## post-warmup draws per chain=2000, total post-warmup draws=8000.
##
##            mean se_mean      sd    2.5%     25%     50%     75%   97.5% n_eff
## pis[1]  0.22406 0.00035 0.04600 0.14135 0.19184 0.22185 0.25312 0.31968 17048
## pis[2]  0.09914 0.00023 0.02965 0.04841 0.07815 0.09683 0.11711 0.16391 17137
## pis[3]  0.18261 0.00036 0.04864 0.09753 0.14743 0.17973 0.21359 0.28689 18303
## pis[4]  0.19439 0.00034 0.04584 0.11213 0.16171 0.19176 0.22432 0.29126 17975
## pis[5]  0.16161 0.00025 0.03269 0.10329 0.13864 0.15998 0.18338 0.22854 16847
## pis[6]  0.26592 0.00036 0.04886 0.17482 0.23243 0.26451 0.29699 0.36620 18908
## pis[7]  0.17940 0.00029 0.03810 0.11172 0.15255 0.17715 0.20438 0.25802 17246
## pis[8]  0.13758 0.00024 0.03000 0.08481 0.11638 0.13552 0.15728 0.20130 16105
## pis[9]  0.11618 0.00014 0.01793 0.08348 0.10362 0.11541 0.12800 0.15307 17072
## pis[10] 0.46295 0.00034 0.04499 0.37527 0.43174 0.46255 0.49371 0.54955 17080
## r       0.19504 0.00049 0.04728 0.10286 0.16386 0.19519 0.22636 0.28724  9415
## r_mean  0.20238 0.00010 0.01246 0.17821 0.19381 0.20223 0.21065 0.22709 16872
##            Rhat
## pis[1]  0.99973
## pis[2]  0.99978
## pis[3]  0.99971
## pis[4]  0.99977
## pis[5]  0.99969
## pis[6]  0.99993
## pis[7]  0.99973
## pis[8]  0.99976
## pis[9]  0.99972
## pis[10] 0.99974
```

```
## r       0.99992
## r_mean  0.99971
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 12 21:41:09 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
posterior_r <- extract(model.fit2)$r
r_estimate <- mean(posterior_r)
r_estimate
```

```
## [1] 0.1950411
```

```
posterior_r_mean <- extract(model.fit2)$r_mean
r_estimate_mean <- mean(posterior_r_mean)
r_estimate_mean
```

```
## [1] 0.2023835
```

```
pis_test <- extract(model.fit2)$pis
pis_test[1,]
```

```
##  [1] 0.16817427 0.06408609 0.22834076 0.23830615 0.13180650 0.29639547
##  [7] 0.25558838 0.14548326 0.18257751 0.47881650
```

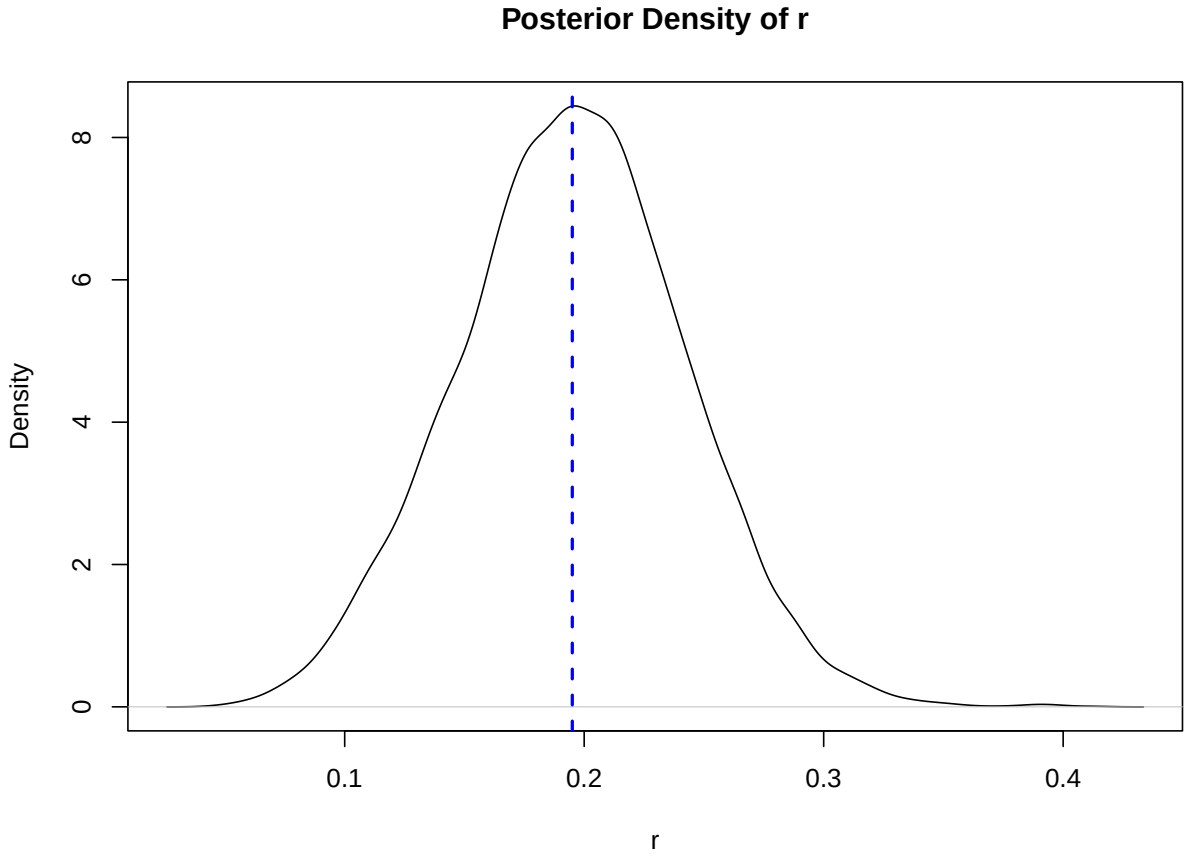
```
posterior_r[1]
```

```
## [1] 0.1338427
```

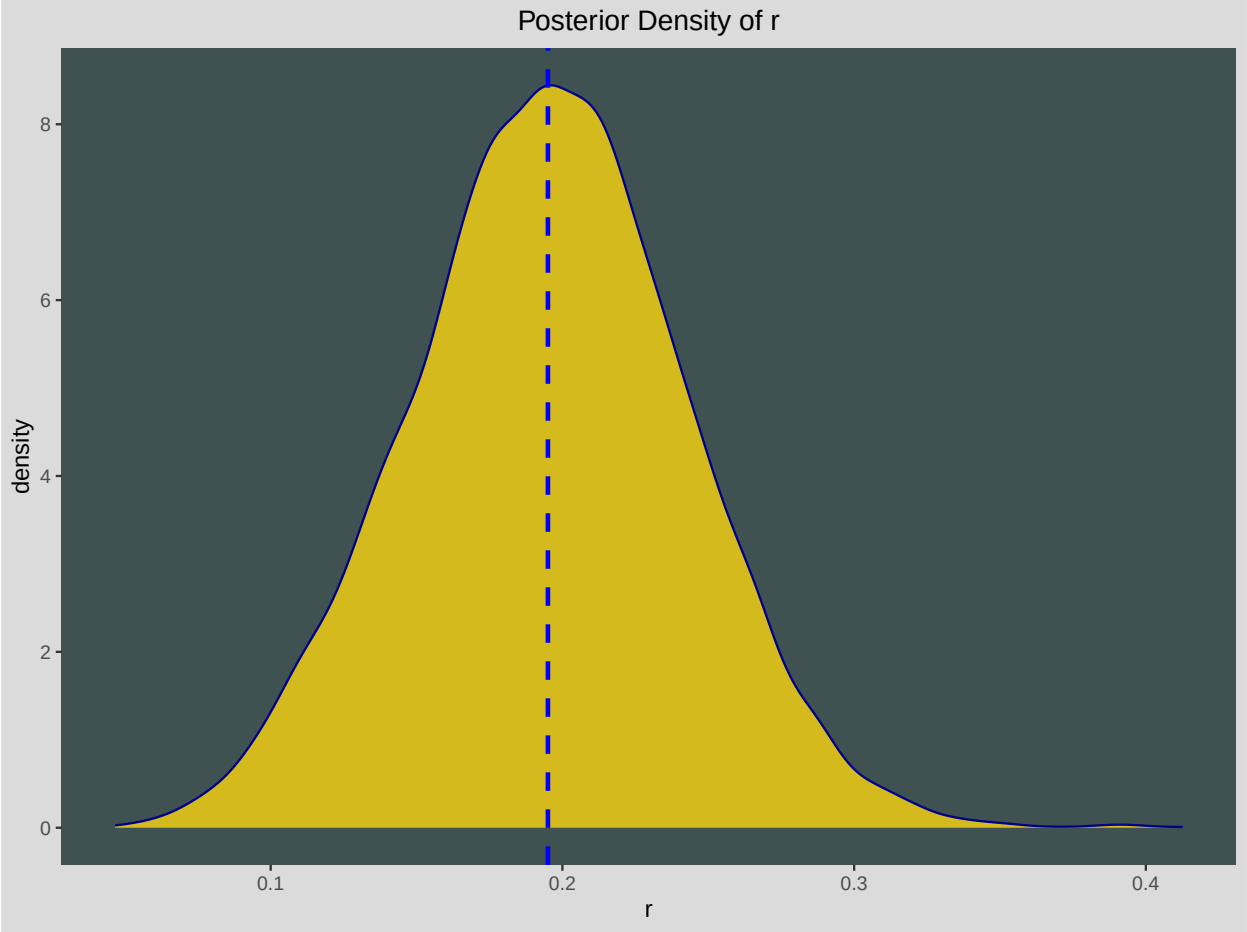
```
get_posterior_mean(model.fit2, pars = "r")
```

```
##   mean-chain:1 mean-chain:2 mean-chain:3 mean-chain:4 mean-all chains
## r    0.1950053    0.1943793     0.194907    0.1958728       0.1950411
```

```
plot(density(posterior_r), main="Posterior Density of r", xlab="r", ylab="Density")
abline(v=r_estimate, col="blue", lwd=2, lty=2)
```




```
data_to_plot_2 <- data.frame(r = posterior_r)
ggplot(data_to_plot_2, aes(x=r)) + geom_density(color="darkblue", fill="#FAD510", alpha = 0.8) + geom_vl
            color="blue", linetype="dashed", linewidth=1) + ggtitle(paste("Posterior Density of r"))  +
theme(plot.title = element_text(hjust = 0.5),panel.grid.major = element_blank(), panel.grid.minor = eler
```




```
#plot(density(posterior_r_mean), main="Posterior Density of r", xlab="r", ylab="Density")
#abline(v=r_estimate_mean, col="blue", lwd=2, lty=2)
```