Assign_3

23370209_Franco Meng

2024-10-29

```
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()
```

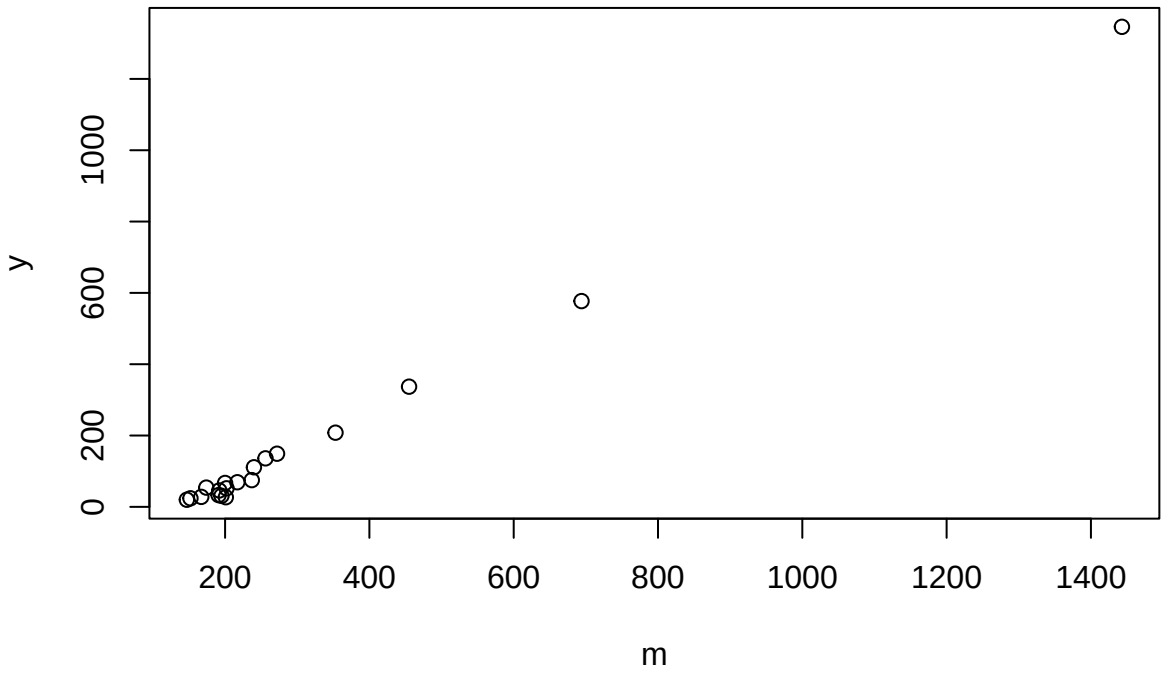
```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

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

TASK 1

```r
dat <- read.table("Golf.csv", header=TRUE, sep = ",")
plot(y~m, dat)
```



Qa : Because the involvement with log in the logit function, here we consider factor change in X (distance from the hole), rather than talking about X+1

I have attached a handwritten calculation for this part.

The odds of success (hitting into the hole) from a distance of a factor 'c' times distance x , are C^Beta1 times the odd of success from the distance at x.

Qb :

```
data{
  int<lower=1> n;
  int<lower=1> p;
  matrix[n, p] x;
  int<lower=0> y[n];
  int<lower=0> mi[n];
}

transformed data{
  matrix[p, p] R = qr_thin_R(log(x));
  real s = sqrt(n - 1.0);
  matrix[p, p] R_ast = R/s;
  matrix[n, p] Q_ast = qr_thin_Q(log(x))*s;
  matrix[p, p] R_ast_inverse = inverse(R_ast);
```

```
  }

  parameters{
    vector[p] theta;  // regression coefficients for predictors
  }

  model{
     // likelihood
     y  ~ binomial_logit(mi, Q_ast*theta);

     // priors
     // Stan puts automatically flat priors on the thetas
  }

  generated quantities{
    vector[p] beta = R_ast_inverse * theta;
    real T = 0 ;
    real y_rep[n];
    real T_rep = 0;
    y_rep = binomial_rng(mi, inv_logit(Q_ast*theta));
    for(i in 1:n){
         T += (y[i] - mi[i] * inv_logit(Q_ast*theta)[i] )^2 / (mi[i] * inv_logit(Q_ast*theta)[i] *(1-inv
         T_rep += (y_rep[i] - mi[i] * inv_logit(Q_ast*theta)[i] )^2 / (mi[i] * inv_logit(Q_ast*theta)[i]

    }
  }
```

```
mi <- dat$m
y <- dat$y
x <- cbind(exp(1), dat$distance)
data.in <- list(x=x, mi=mi, y=y, n=NROW(x), p=NCOL(x))
model.fit <- sampling(BinomialLogitGLMQR, data=data.in,iter=10000, warmup = 2000)
```

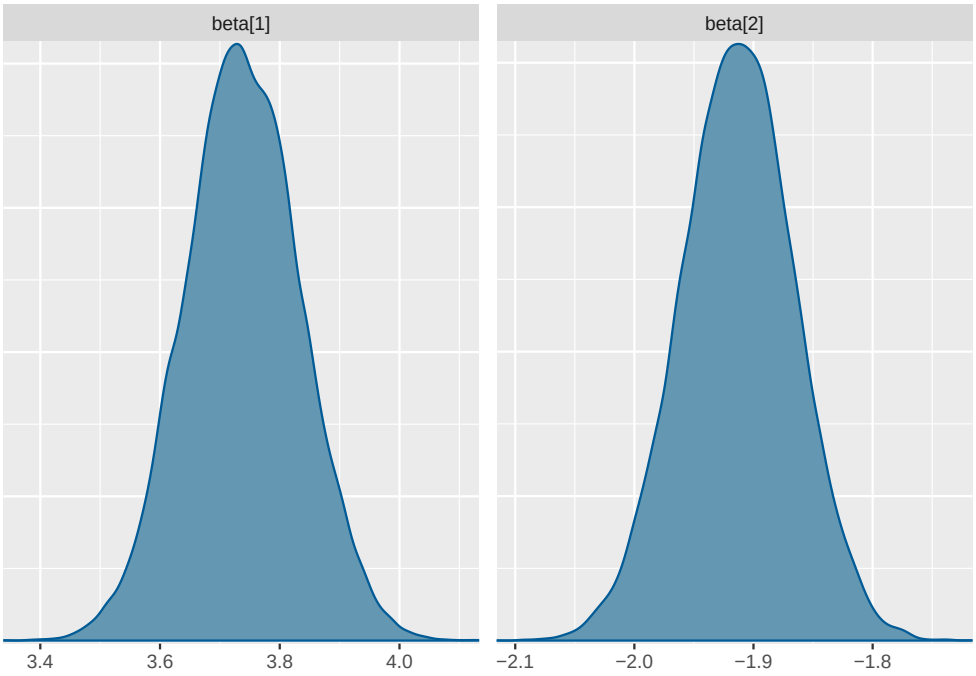
```
print(model.fit, digits=5, pars="beta")
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=2000; thin=1;
## post-warmup draws per chain=8000, total post-warmup draws=32000.
##
##             mean se_mean      sd     2.5%      25%     50%      75%    97.5%
## beta[1]  3.73825 0.00069 0.09547  3.55269  3.67381  3.7372  3.80263  3.92697
## beta[2] -1.91331 0.00039 0.04738 -2.00658 -1.94505 -1.9130 -1.88150 -1.82062
##         n_eff    Rhat
## beta[1] 19181 1.00036
## beta[2] 14943 1.00057
##
## Samples were drawn using NUTS(diag_e) at Sat Nov  2 22:07:52 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).
```

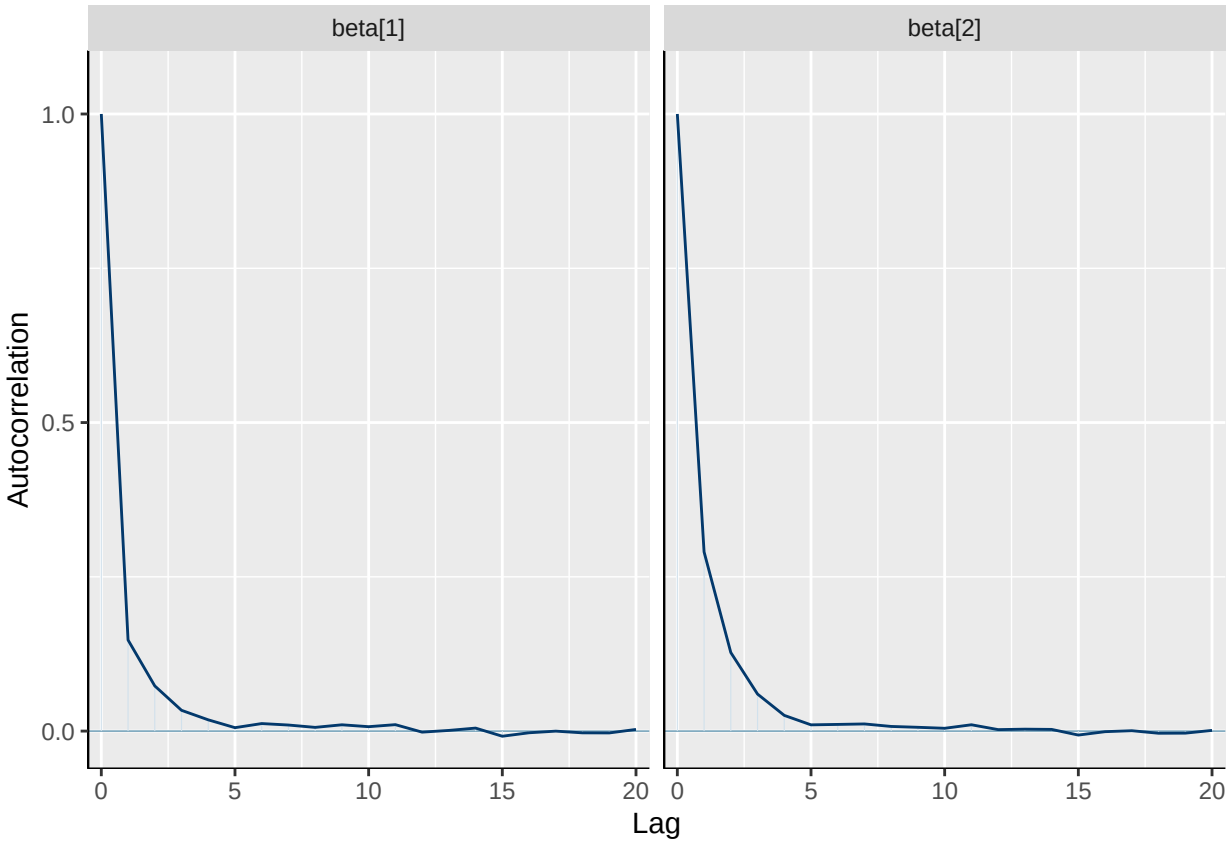
```
check_hmc_diagnostics(model.fit)
```

```
##
## Divergences:

## 0 of 32000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 32000 iterations saturated the maximum tree depth of 10.

##
## Energy:

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

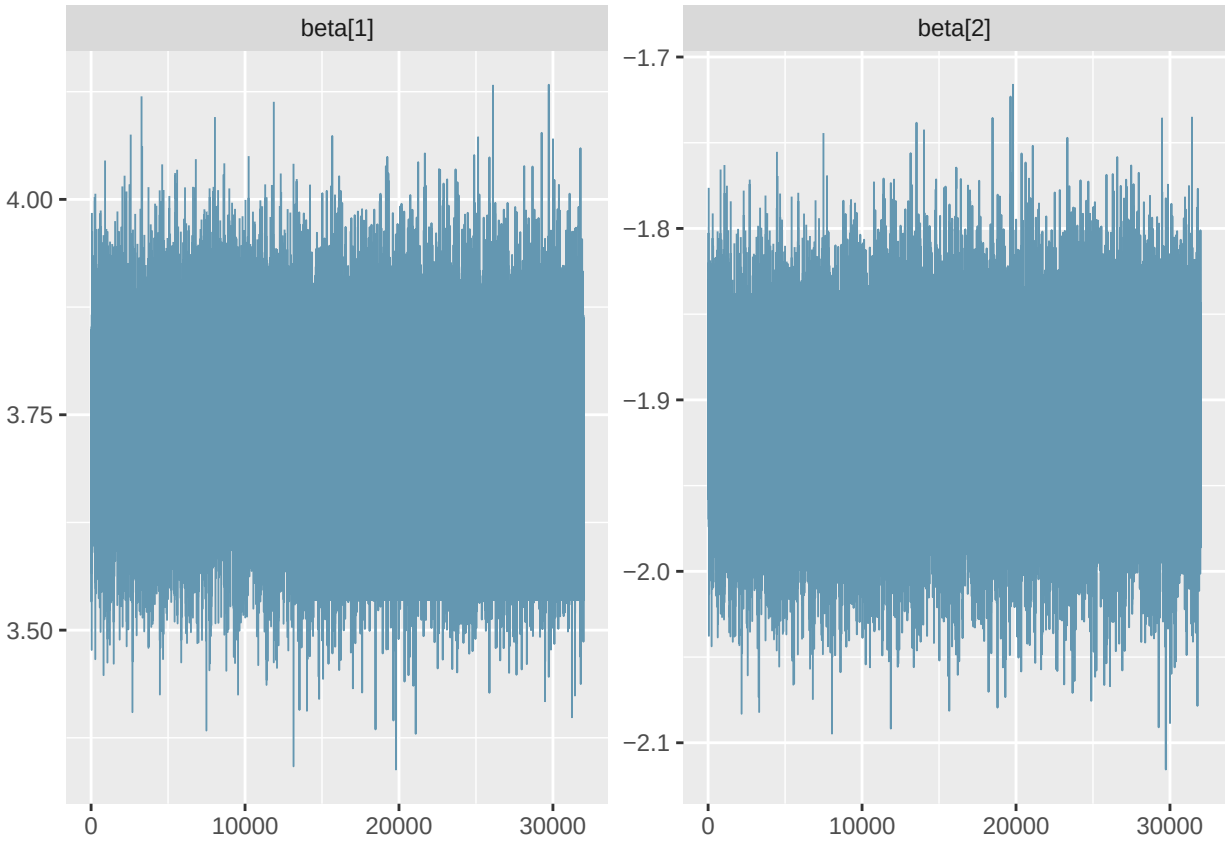
```
posterior <- as.matrix(model.fit)
mcmc_dens(posterior, regex_pars = "beta")
```




```
mcmc_acf(posterior,regex_pars = "beta")
```




```
mcmc_trace(posterior, regex_pars = "beta")
```



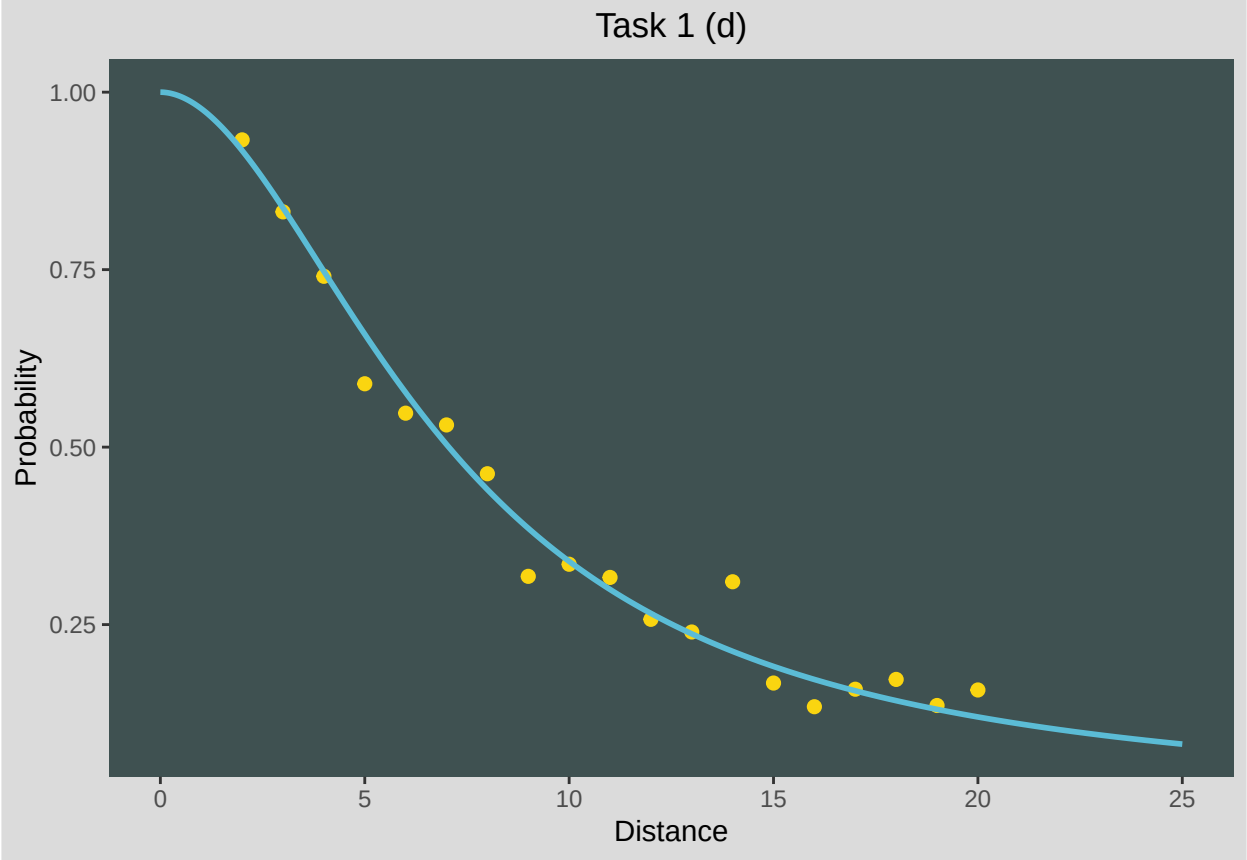

```
##plot(y/m ~ distance, dat)
##beta <- colMeans(extract(model.fit, "beta")[[1]])
##curve(1/(1+exp(-(beta[1]+beta[2]*log(x)))), from = 0, to = 25, n=301, add=TRUE)
```

```
 fit1d <- colMeans(extract(model.fit, "beta")[[1]])
 names(fit1d) <- c("Beta0", "Beta1")
 xgr <- with(dat, seq(from = 0, to = 25, length = 601))
 line_to_plot <- data.frame(distance = xgr, ob_successful_proportion = 1/(1+exp(-(fit1d["Beta0"]+fit1d[
```

```
data_to_plot <- data.frame(distance = dat$distance, ob_successful_proportion = dat$y/dat$m)
#data_to_plot
ggplot() +
  geom_point(data = data_to_plot, aes(x = distance, y= ob_successful_proportion), color = "#FAD510", si
  geom_line(data = line_to_plot, aes(x=distance, y =ob_successful_proportion ), color = "#5BBCD6", size
  ggtitle("Task 1 (d)") +
  labs(x = "Distance", y = "Probability") +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "#3F5151"),
        plot.background = element_rect(fill = "gray86"))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
```

```
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```




```
test <- extract(model.fit, "y_rep")

mean(extract(model.fit)$T<extract(model.fit)$T_rep)
```

```
## [1] 0.00946875
```

```
BAR <- read.csv("bicycles.csv")
BAR$x1 <- as.numeric(BAR$BikeRoute == "yes")
BAR$x2 <- as.numeric(BAR$Type == "FairlyBusy")
BAR$x3 <- as.numeric(BAR$Type == "Busy")
BAR$mi <- BAR$Bicycles + BAR$Other
```

```
data{
  int<lower=0> n;              // number of observations
  int<lower=0> m[n];           // the number of trials
  int<lower=0> y[n];
  matrix[n,5] X;
}
```

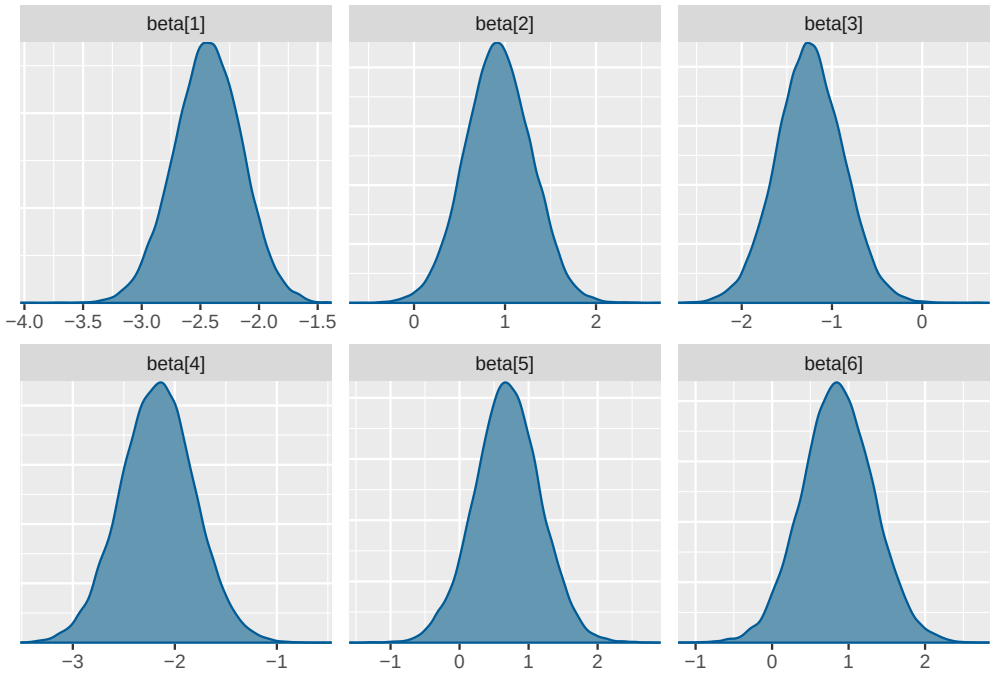
```
parameters{
  vector[6] beta;
  real<lower=0> sigma_alpha;
  vector[n] alpha0;
  //vector[n] logit_theta;
}

transformed parameters{
  vector[n] theta;
  for (i in 1:n)
  theta[i] = inv_logit(beta[1] + alpha0[i] + X[i, 1] * beta[2] + X[i, 2] * beta[3] + X[i, 3] * beta[4] +
}

model{
  y ~ binomial(m, theta); // Binomial likelihood with logit link
  beta ~ normal(0, 100);          // priors for beta1 to beta5
  alpha0 ~ normal(0, sigma_alpha); // random intercepts
  sigma_alpha ~ cauchy(0, 5);
}


generated quantities {
  real y_pred[n];                      // Posterior predictive distribution for each observation
  vector[n] theta_new;
  theta_new = inv_logit(alpha0 + beta[1] + 1 * beta[4]) ;
  y_pred = binomial_rng(200, theta_new);  // Predictive distribution with m = 200

}
```

```
mi <- BAR$m
y <- BAR$Bicycles
x <- cbind(BAR$x1, BAR$x2, BAR$x3, BAR$x1*BAR$x2, BAR$x1*BAR$x3)
data.in_2 <- list(X=x, m=mi, y=y, n=NROW(x))
model.fit_2 <- sampling(task2, data=data.in_2, iter=10000, warmup = 2000)
```

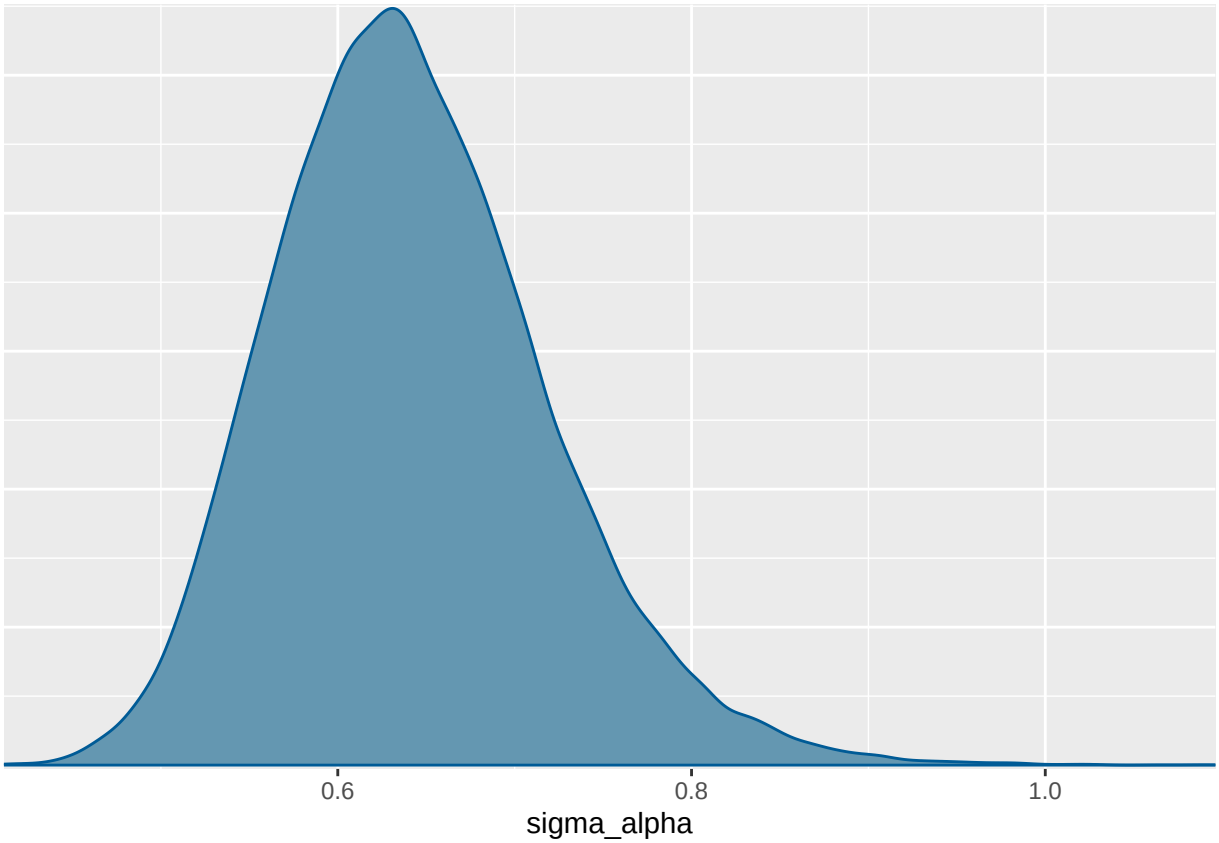
```
check_hmc_diagnostics(model.fit_2)
```

```
##
## Divergences:

## 0 of 32000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 32000 iterations saturated the maximum tree depth of 10.

##
## Energy:

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

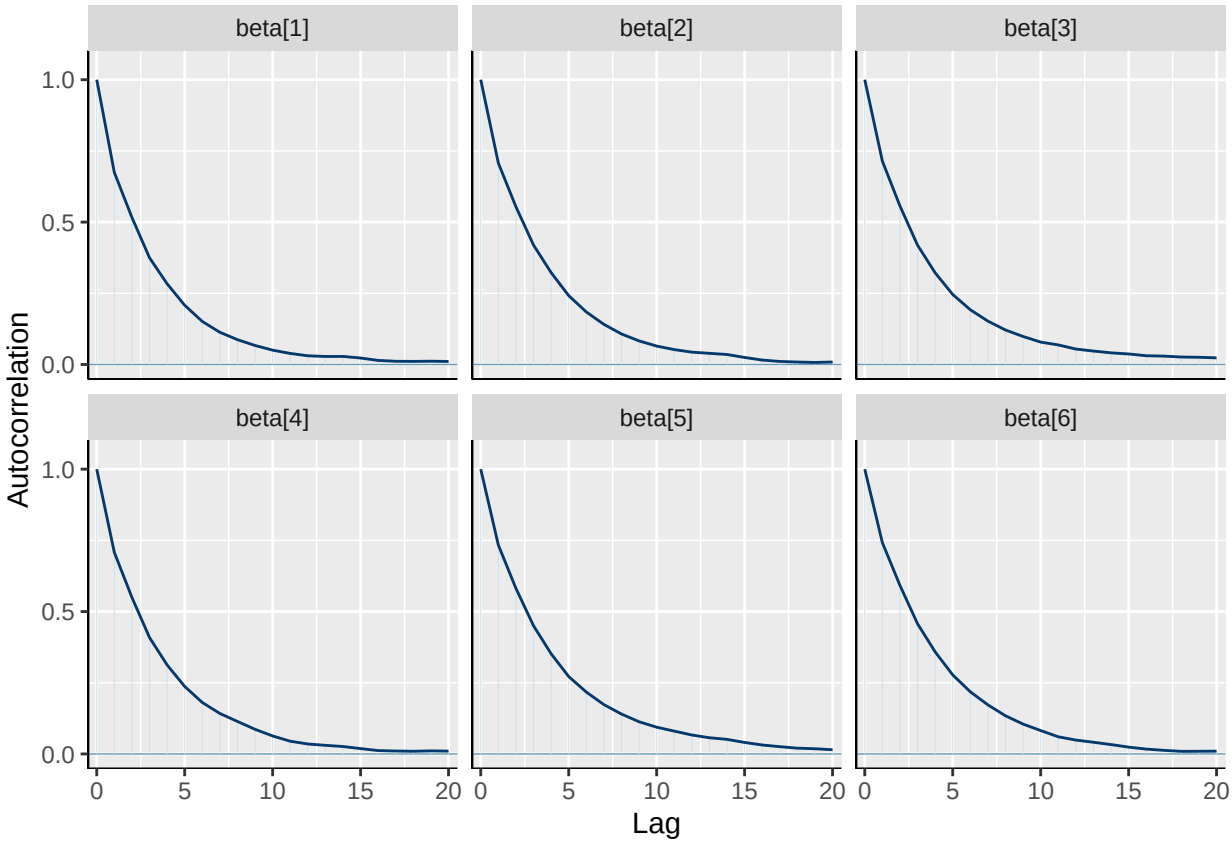
```
posterior_2 <- as.matrix(model.fit_2)
mcmc_dens(posterior_2, regex_pars = "beta")
```



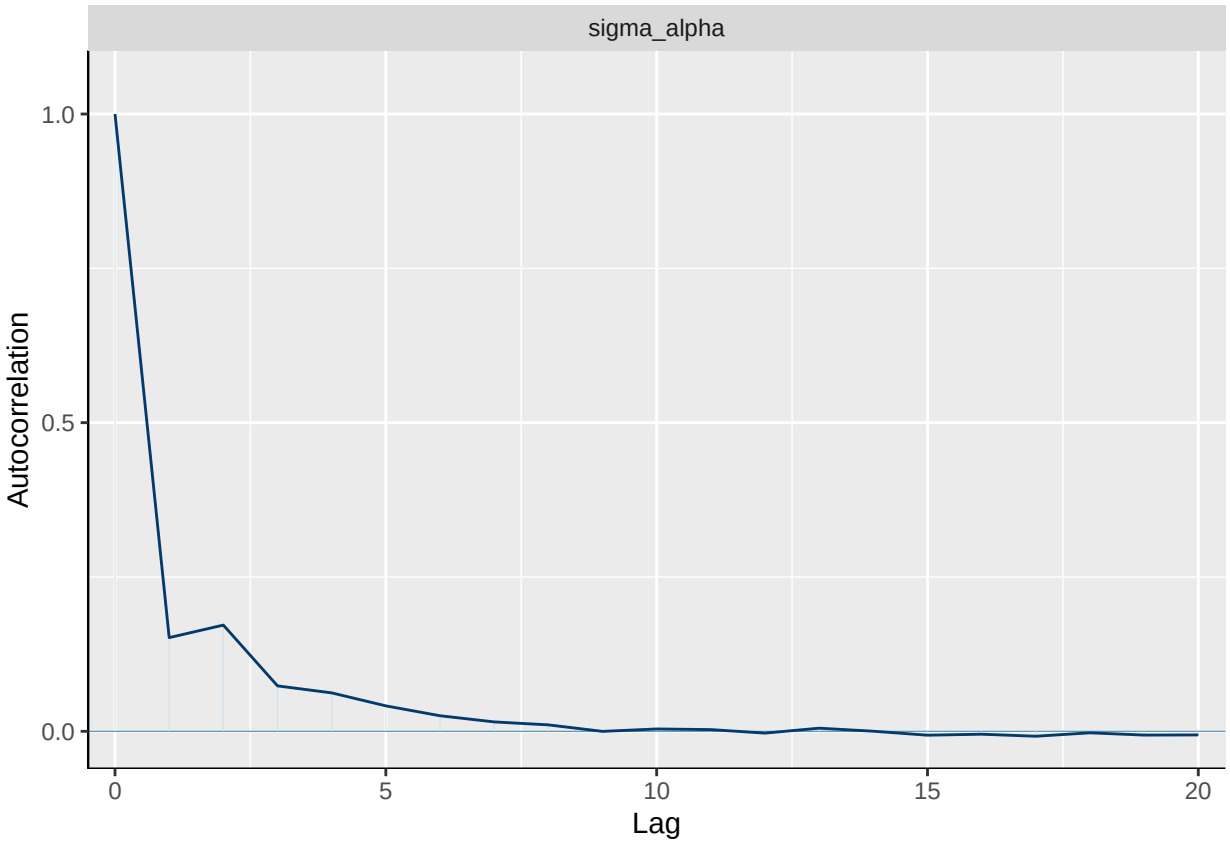

```
mcmc_dens(posterior_2, pars = c("sigma_alpha"))
```



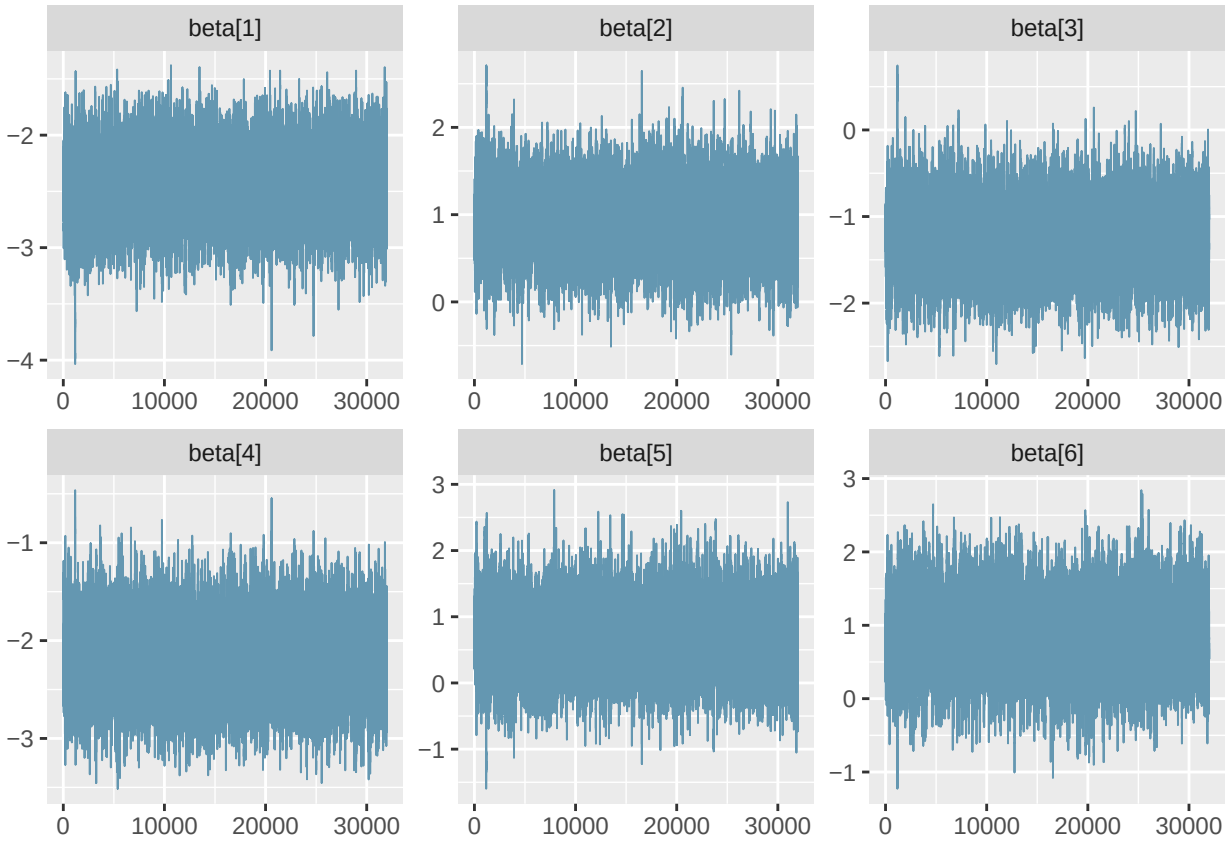

```
mcmc_acf(posterior_2,regex_pars = c("beta"))
```




```
mcmc_acf(posterior_2,pars = "sigma_alpha")
```




```
mcmc_trace(posterior_2,regex_pars = c("beta"))
```




```
print(model.fit_2, digits=5, pars=c("beta","sigma_alpha"))
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=2000; thin=1;
## post-warmup draws per chain=8000, total post-warmup draws=32000.
##
##                mean se_mean      sd     2.5%      25%      50%      75%
## beta[1]     -2.43618 0.00415 0.29044 -3.01146 -2.62869 -2.43345 -2.23859
## beta[2]      0.92728 0.00548 0.36457  0.21546  0.68300  0.92422  1.17096
## beta[3]     -1.24314 0.00589 0.37135 -1.96666 -1.49074 -1.24700 -0.99693
## beta[4]     -2.16789 0.00544 0.36457 -2.88828 -2.41208 -2.16641 -1.92575
## beta[5]      0.69422 0.00772 0.48021 -0.26321  0.37891  0.69193  1.01028
## beta[6]      0.85931 0.00746 0.47111 -0.05533  0.54810  0.85570  1.17337
## sigma_alpha  0.64168 0.00062 0.07656  0.50967  0.58808  0.63593  0.68846
##                97.5% n_eff    Rhat
## beta[1]     -1.87572  4907 1.00150
## beta[2]      1.63571  4430 1.00123
## beta[3]     -0.51274  3975 1.00173
## beta[4]     -1.44650  4498 1.00189
## beta[5]      1.63724  3872 1.00138
## beta[6]      1.78803  3988 1.00179
## sigma_alpha  0.80910 15099 0.99999
##
## Samples were drawn using NUTS(diag_e) at Sat Nov  2 22:10:00 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).
```

```
fit2d <- c(mean(colMeans(extract(model.fit_2, "alpha0")[[1]])), colMeans(extract(model.fit_2, "beta")[[
busy_bikeroute    <- c(1,1,1,0,1,0,1)
busy_no_bikeroute <- c(1,1,0,0,1,0,0)
sum(fit2d*busy_bikeroute)
```

```
## [1] -2.81587
```

```
sum(fit2d*busy_no_bikeroute)
```

```
## [1] -4.602459
```

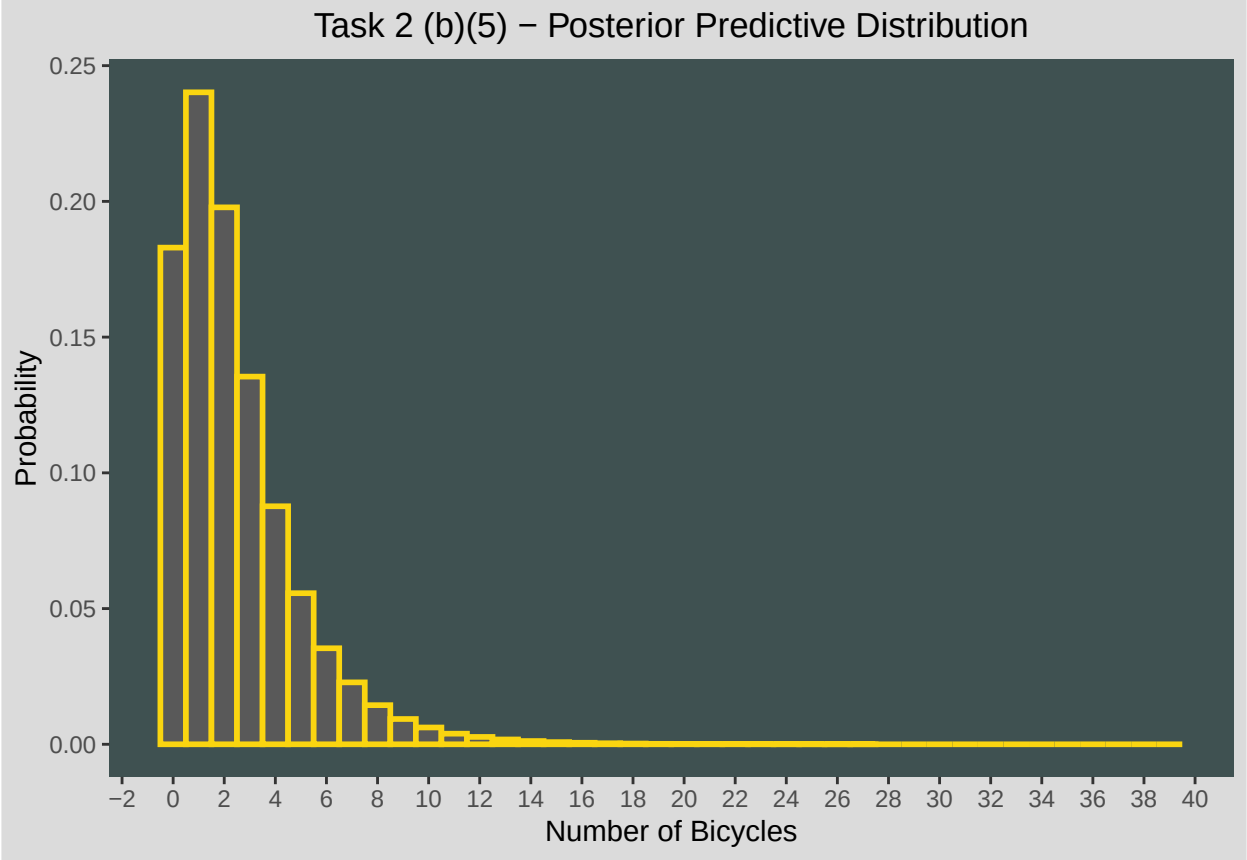
```
exp(sum(fit2d*busy_bikeroute)) / exp (sum(fit2d*busy_no_bikeroute))
```

```
## [1] 5.969059
```

```
exp(fit2d[3]+fit2d[7])
```

```
## [1] 5.969059
```

```
data_to_plot <- data.frame(new = as.vector(extract(model.fit_2, "y_pred")[[1]]))
ggplot() +
  geom_histogram(data = data_to_plot, aes(x = new , y = after_stat(density)), color = "#FAD510", size =
  ggtitle("Task 2 (b)(5) - Posterior Predictive Distribution") +
  labs(x = "Number of Bicycles", y = "Probability")+   scale_x_continuous(n.breaks=30) +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "#3F5151"),
        plot.background = element_rect(fill = "gray86"))
```




```
test_3 <- extract(model.fit_2, "beta[1]")[[1]] + extract(model.fit_2, "beta[4]")[[1]]
p <- exp(test_3) / (1+exp(test_3))
rbinom(1,200,p)
```

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

TASK 3

```
dat.full <- read.csv("pregnancies.csv")
```

```
NS <- sum(dat.full$Smoker)
NNS <- sum(dat.full$Nonsmoker)
n <- length(dat.full$Nonsmoker)
yS <- dat.full$Smoker
yNS <- dat.full$Nonsmoker
data.in <- list(n = n, NS = NS, NNS = NNS, yS = yS, yNS=yNS)
```

```
  data {
  int<lower=0> NS;
  int<lower=0> NNS;
  int <lower=1> n;
  int<lower=0> yS[n];
```

```
  int<lower=0> yNS[n];
}

parameters {
  real<lower=0> muS;
  real<lower=0> thetaS;
  real<lower=0> muNS;
  real<lower=0> thetaNS;
}

transformed parameters {
  real<lower=2> alphaS = 2 + 2 * thetaS;
  real<lower=0> betaS = muS * (1 + 2 * thetaS);
  real<lower=2> alphaNS = 2 + 2 * thetaNS;
  real<lower=0> betaNS = muNS * (1 + 2 * thetaNS);

  simplex[n] probspiS; // special data structure in Stan to hold values that
  simplex[n] probspiNS; // special data structure in Stan to hold values that
  real sumprobpiS = 0;
  real sumprobpiNS = 0;
  probspiS[1] = alphaS/(alphaS+betaS);// probability of being in first category
  probspiNS[1] = alphaNS/(alphaNS+betaNS);// probability of being in first category
  sumprobpiS += probspiS[1];
  sumprobpiNS += probspiNS[1];
  for(i in 2:(n-1)){
    probspiS[i] = probspiS[i-1] * (betaS+i-2)/(betaS+alphaS+i-1); // this is different to the formula i
    probspiNS[i] = probspiNS[i-1] * (betaNS+i-2)/(betaNS+alphaNS+i-1);
    sumprobpiS += probspiS[i];
    sumprobpiNS += probspiNS[i];
    }
    probspiS[n] = 1 - sumprobpiS; // probability of being in last category
    probspiNS[n] = 1 - sumprobpiNS;

  }


model {
  // Priors
  muS ~ exponential(0.1);
  thetaS ~ exponential(0.1);
  muNS ~ exponential(0.1);
  thetaNS ~ exponential(0.1);

  // Likelihoods using multinomial
  yS ~ multinomial(probspiS);
  yNS ~ multinomial(probspiNS);
}

generated quantities{
    real TS = 0 ;
    real TNS = 0 ;
    real ys_rep[n];
    real yns_rep[n];
```

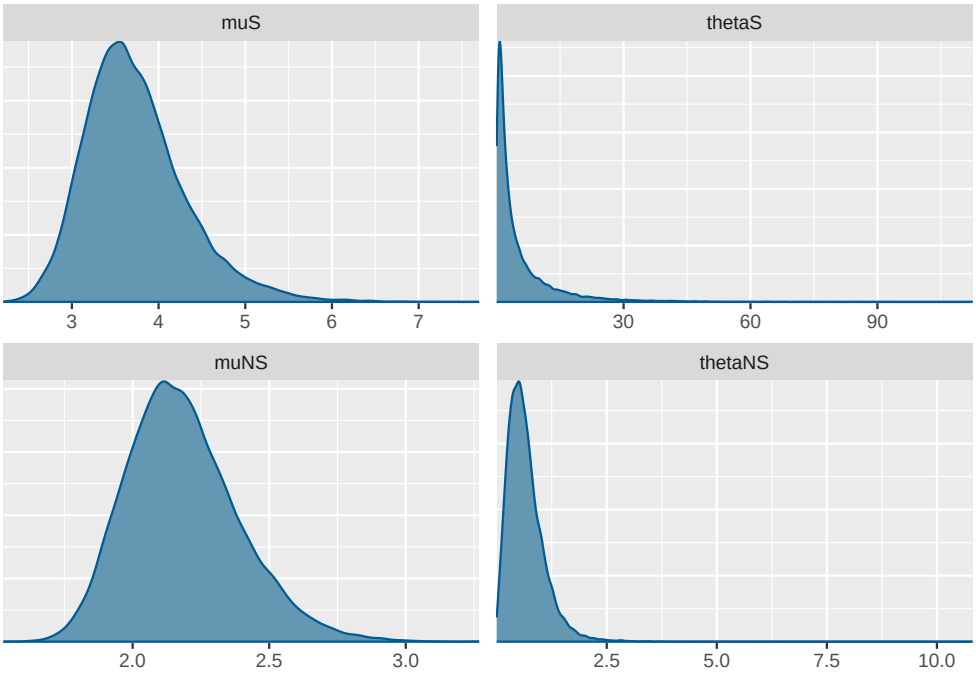
```
    real TS_rep = 0;
    real TNS_rep = 0;
    int ps;
    int pns;
    real probpiS_new;
    real probpiNS_new;
    int ys_new;
    int yns_new;
    ys_rep = multinomial_rng(probspiS,NS);
    yns_rep = multinomial_rng(probspiNS,NNS);
    for(i in 1:n){
        TS += (yS[i]-NS*probspiS[i])^2 / (NS*probspiS[i]*(1-probspiS[i]));
        TNS += (yNS[i]-NNS*probspiNS[i])^2 / (NNS*probspiNS[i]*(1-probspiNS[i]));
        TS_rep += (ys_rep[i]-sum(ys_rep)*probspiS[i])^2 / (sum(ys_rep)*probspiS[i]*(1-probspiS[i]));
        TNS_rep += (yns_rep[i]-sum(yns_rep)*probspiNS[i])^2 / (sum(yns_rep)*probspiNS[i]*(1-probspiNS[i
      }
    ps = TS>TS_rep;
    pns = TNS>TNS_rep;

    probpiS_new = beta_rng(alphaS, betaS);
    probpiNS_new = beta_rng(alphaNS, betaNS);

    ys_new = neg_binomial_rng(1, probpiS_new/(1.0-probpiS_new) ) + 1 ;  // as in this unit Geometric di
    yns_new = neg_binomial_rng(1, probpiNS_new/(1.0-probpiNS_new) ) + 1 ;

}
```

```
fit4 <- sampling(task3_final, data = data.in, iter = 10000, warmup = 2000)
```

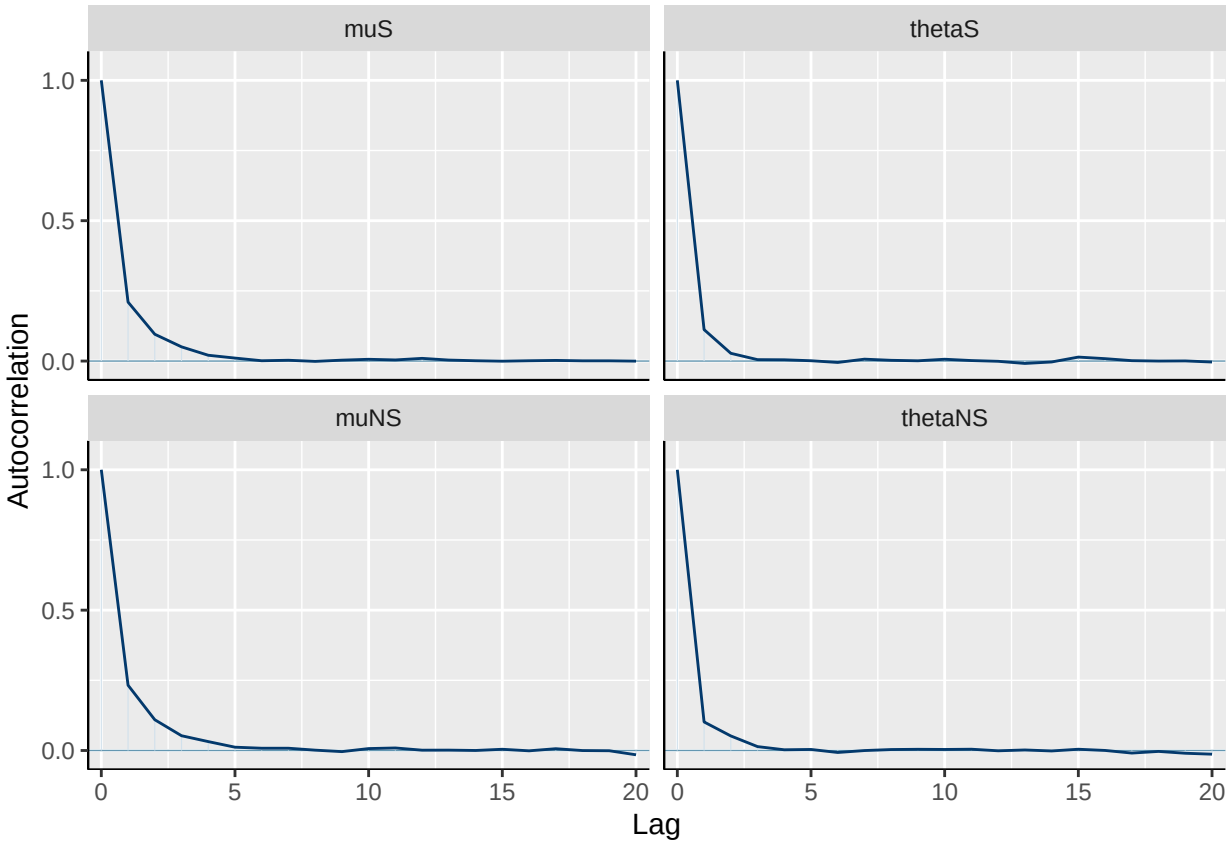
```
check_hmc_diagnostics(fit4)
```

```
##
## Divergences:

## 0 of 32000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 32000 iterations saturated the maximum tree depth of 10.

##
## Energy:

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

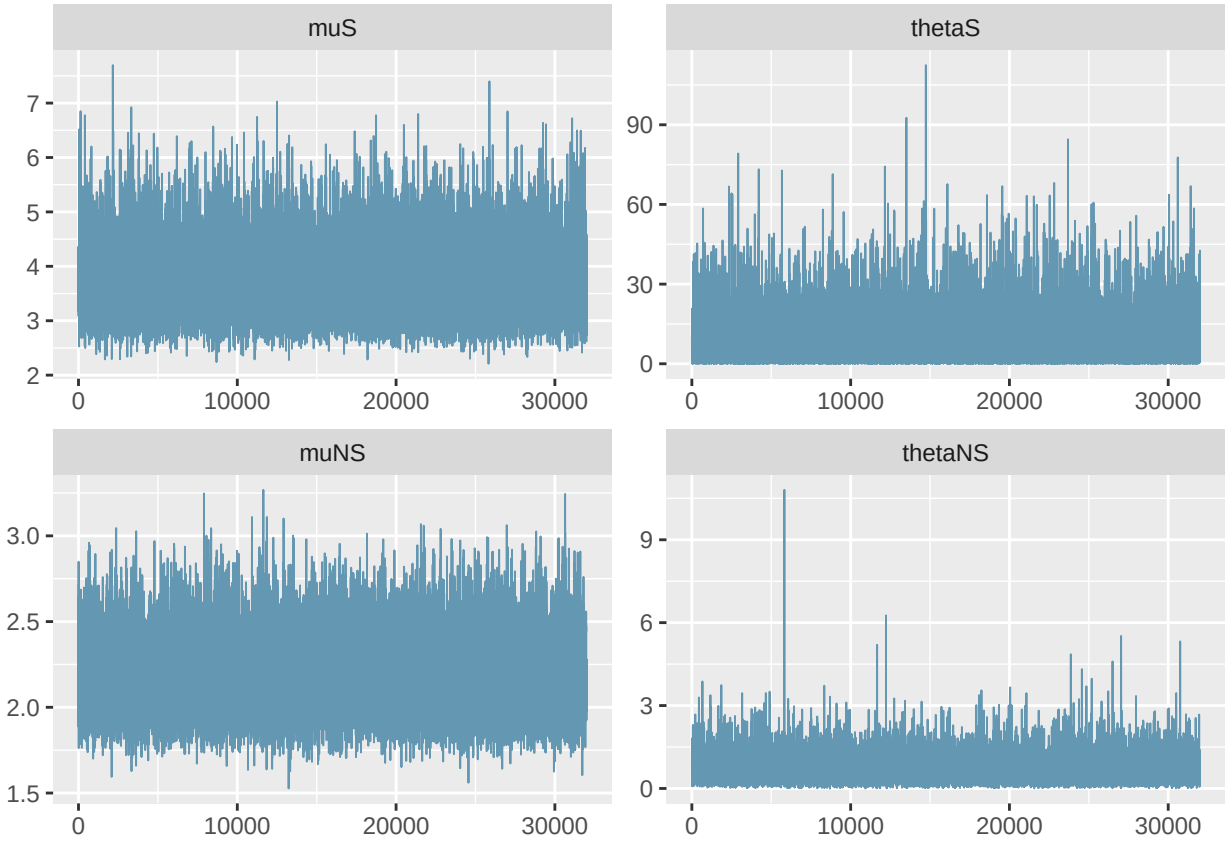
```
posterior4 <- as.matrix(fit4)
mcmc_dens(posterior4, pars = c("muS", "thetaS", "muNS","thetaNS") )
```




```
mcmc_acf(posterior4,pars = c("muS", "thetaS", "muNS","thetaNS"))
```



```
mcmc_trace(posterior4,pars = c("muS", "thetaS", "muNS","thetaNS"))
```



Question (a)

```
print(fit4, pars = c("muS", "thetaS", "muNS","thetaNS"), digits = 5)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=2000; thin=1;
## post-warmup draws per chain=8000, total post-warmup draws=32000.
##
##            mean se_mean      sd    2.5%     25%     50%     75%    97.5% n_eff
## muS     3.73495 0.00439 0.58562 2.80172 3.32914 3.65629 4.05322  5.12665 17794
## thetaS  5.33092 0.04817 7.53346 0.09828 0.91073 2.44993 6.53364 27.36888 24456
## muNS    2.18522 0.00159 0.20532 1.83874 2.04059 2.16625 2.30915  2.64186 16760
## thetaNS 0.65894 0.00275 0.42321 0.09688 0.36703 0.58150 0.85721  1.67381 23758
##            Rhat
## muS     1.00000
## thetaS  0.99991
## muNS    1.00014
## thetaNS 1.00004
##
## Samples were drawn using NUTS(diag_e) at Sat Nov  2 22:12:27 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).
```

Question (b)

```
mean(extract(fit4, "muS")[[1]]-extract(fit4, "muNS")[[1]] > 1)
```

```
## [1] 0.8240625
```

Question (c)

```
mean(extract(fit4, "ps")[[1]])
```

```
## [1] 0.8385938
```

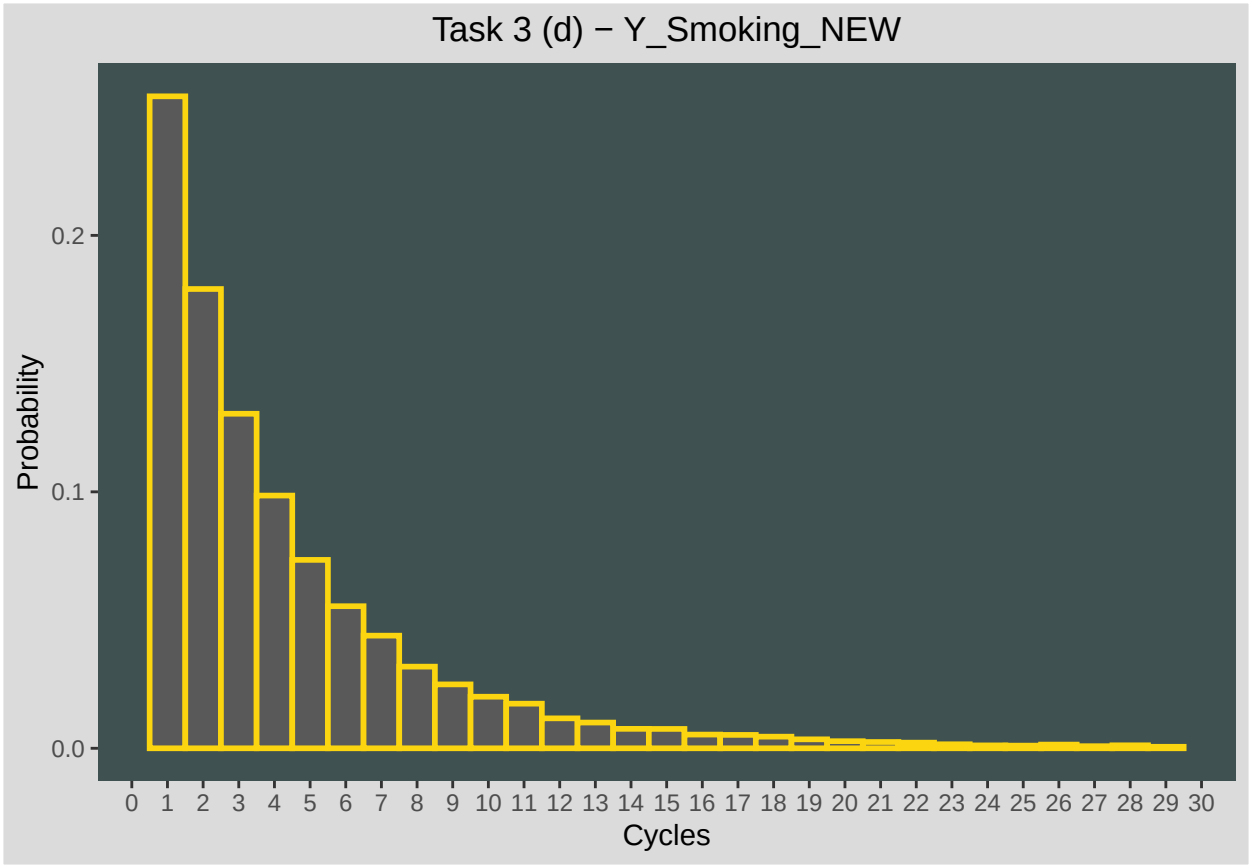
```
mean(extract(fit4, "pns")[[1]])
```

```
## [1] 0.6891875
```

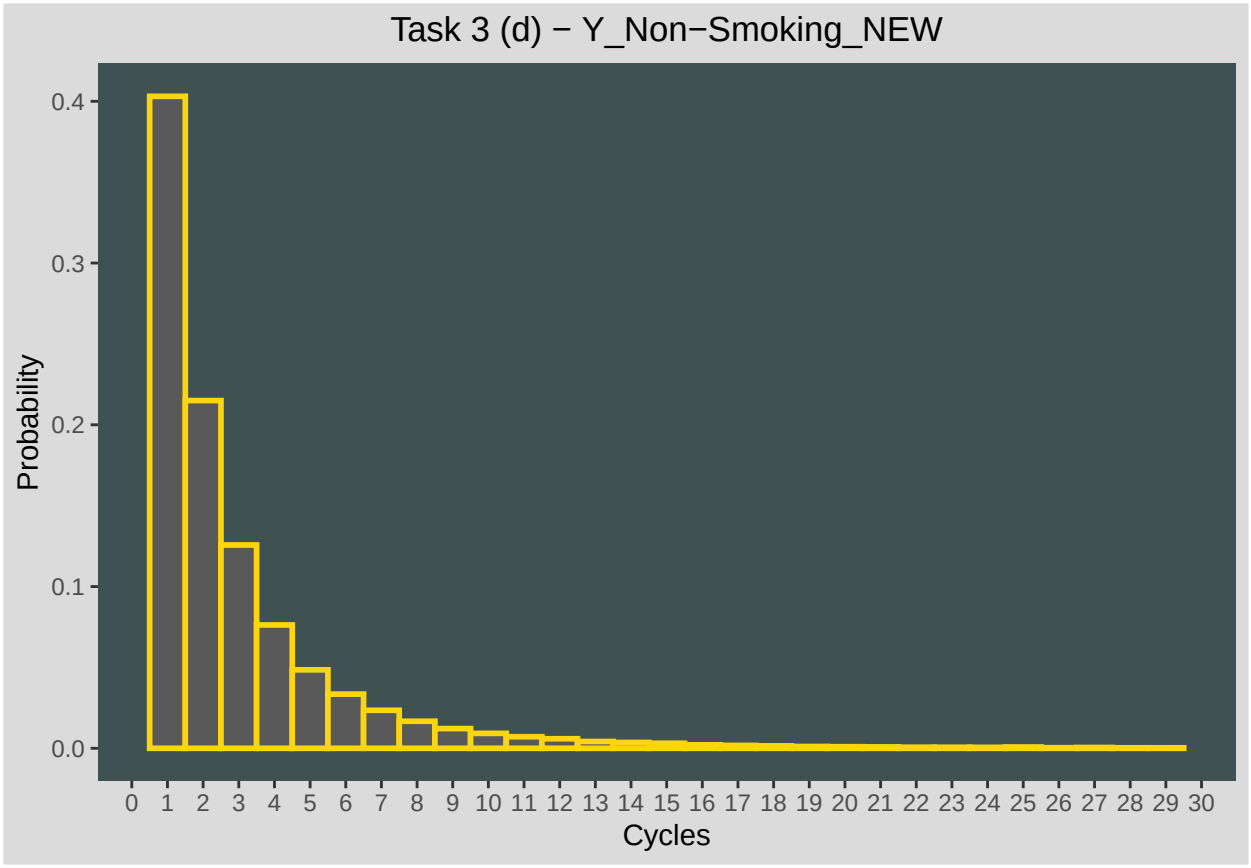
Question (d)

```
plot1 <- extract(fit4, "ys_new")[[1]]
plot2 <- extract(fit4, "yns_new")[[1]]
plot1 <- plot1[plot1<30]
plot2 <- plot2[plot2<30]
```

```
data_to_plot <- data.frame(ys_new = plot1)
ggplot() +
  geom_histogram(data = data_to_plot, aes(x = ys_new, y = after_stat(density)), color = "#FAD510", size
  ggtitle("Task 3 (d) - Y_Smoking_NEW") +
  labs(x = "Cycles", y = "Probability")+   scale_x_continuous(n.breaks=30) +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "#3F5151"),
        plot.background = element_rect(fill = "gray86"))
```




```
data_to_plot <- data.frame(ys_new = plot2)
ggplot() +
  geom_histogram(data = data_to_plot, aes(x = ys_new, y = after_stat(density)), color = "#FAD510", size
  ggtitle("Task 3 (d) - Y_Non-Smoking_NEW") +
  labs(x = "Cycles", y = "Probability") +   scale_x_continuous(n.breaks=30) +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "#3F5151"),
        plot.background = element_rect(fill = "gray86"))
```


Task 3 (d) − Y_Non−Smoking_NEW
Probability
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Cycles