```r
# Load the lars package and the diabetes dataset
library(reshape2)
library(lars)
data(diabetes)
library(GGally)

library(ggplot2)
library(gridExtra)

library("rstan")
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
#data
X_matrix <- diabetes$x
class(X_matrix) <- "matrix"
y_vector <- diabetes$y

X_design <- cbind(1, X_matrix)

data.raw <- read.table(file='https://hastie.su.domains/Papers/LARS/diabetes.data', header=T)
```

```r
#eda 1
plot1 <- ggpairs(data.raw[c(1,3,4,11)],
    upper=list(continuous=wrap("cor", size=4)),
    progress = FALSE)

ggsave("./plots/eda_all_p1.pdf", plot1,  width = 10, height = 8)
plot2 <- ggpairs(data.raw[c(5:10,11)],
    upper=list(continuous=wrap("cor", size=4)),
    progress = FALSE)

ggsave("./plots/eda_all_p2.pdf", plot2,  width = 10, height = 8)
```

```r
#eda 2
plot1 <- ggpairs(data.raw, columns=c(1,3,4,11),
    ggplot2::aes(color=factor(SEX), alpha=0.7),
    upper=list(continuous=wrap("cor", size=4)),
    progress = FALSE)
ggsave("./plots/eda_sex_p1.pdf", plot1,  width = 10, height = 8)
plot2 <- ggpairs(data.raw, columns=5:11,
    ggplot2::aes(color=factor(SEX), alpha=0.7),
    upper=list(continuous=wrap("cor", size=4)),
    progress = FALSE)
ggsave("./plots/eda_sex_p2.pdf", plot2,  width = 10, height = 8)
```

```r
#eda 3
cormat <- round(cor(data.raw[1:11]), 2)

get_upper_tri <- function(cormat){
    cormat[lower.tri(cormat)]<- NA
    return(cormat)
}

upper_tri <- get_upper_tri(cormat)
```

```r
# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)

# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat,
                    aes(Var2, Var1, fill = value)) +
             geom_tile(color = "white")+
             scale_fill_gradient2(low = "blue", high = "red",
                                  mid = "white", midpoint = 0,
                                  limit = c(-1,1), space = "Lab",
                                  name="Pearson\nCorrelation") +
             theme_minimal()+ # minimal theme
             theme(axis.text.x=element_text(angle=45, vjust = 1,
                                            size=12, hjust=1)) +
             coord_fixed()

p <- ggheatmap +
     geom_text(aes(Var2, Var1, label = value),
          color = "black", size = 4) +
     theme(
     axis.title.x = element_blank(),
     axis.title.y = element_blank(),
     panel.grid.major = element_blank(),
     panel.border = element_blank(),
     panel.background = element_blank(),
     axis.ticks = element_blank(),
     legend.justification = c(1, 0),
     legend.position = c(0.6, 0.7),
     legend.direction = "horizontal")+
     guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
               title.position = "top", title.hjust = 0.5))
ggsave("./plots/eda_corr_heatmap.pdf", p)
```

```r
#model m1 default bayesian regression
set.seed(123)
# Create a data list for Stan
data_list_m1 <- list(
  N = length(y_vector),  # Number of observations
  K = dim(X_design)[2], # Number of predictors
  x = X_design,  # Predictor variable
  y = y_vector  # Response variable
)


# Compile the Stan model
stan_m1 <- stan_model(file='./priors/prior_M1_v2.stan')

# Fit the model to the data
stan_fit_m1 <- sampling(stan_m1,
                     data = data_list_m1,
                     chains = 4,
                     iter = 2000)

# Print a summary of the results
```

```r
print(stan_fit_m1)

# Plot the posterior distributions
plot(stan_fit_m1,
     pars=c('beta', 'sigma'))
```

```stan
data {
  int<lower=1> N;   // number of data items
  int<lower=1> K;   // number of predictors, contain intercept
  matrix[N, K] x;   // predictor matrix
  vector[N] y;      // outcome vector
}
parameters {
  vector[K] beta;       // coefficients for predictors
  real<lower=0> sigma;  // standard deviation
}
model {
  //improper prior
  target += -2 * log(sigma);
  // likelihood
  y ~ normal(x * beta, sigma);
}
```

```r
#m2 conjuage prior
# Create a data list for Stan
#set.seed(123)
data_list_m2 <- list(
  N = length(y_vector),  # Number of observations
  K = dim(X_design)[2], # Number of predictors, contain intercept
  x = X_design,  # Predictor variable
  y = y_vector,  # Response variable

  m0 = rep(1, dim(X_design)[2]),
  C0 = diag(1, dim(X_design)[2]),
  v0 = 1,
  s0 = 1
)


# Compile the Stan model
stan_m2 <- stan_model(file='./priors/prior_M2.stan')

# Fit the model to the data
stan_fit_m2 <- sampling(stan_m2,
                     data = data_list_m2,
                     chains = 4,
                     iter = 2000)

# Print a summary of the results
print(stan_fit_m2)

# Plot the posterior distributions
g <- plot(stan_fit_m2,
     pars=c('beta', 'sigma'))
```

```r
ggsave("./plots/M2_conju_prior.pdf", g, width = 8, height = 6)
```

```stan
data {
  int<lower=1> N;   // number of data items
  int<lower=1> K;   // number of predictors, contain intercept
  matrix[N, K] x;   // design matrix
  vector[N] y;      // outcome vector

  // parameters for the priors
  // for beta
  vector[K] m0;
  matrix[K, K] C0;
  //for sigma^2
  real<lower=0> v0;
  real<lower=0> s0;

}
parameters {
  vector[K] beta;       // coefficients for predictors
  real<lower=0> sigma2;  //variance
}
transformed parameters {
  real<lower=0> sigma = sqrt(sigma2);
}
model {
 // prior
 sigma2 ~ inv_gamma(v0, s0);
 beta ~ multi_normal(m0, sigma2 * C0);

 // likelihood
 y ~ normal(x * beta, sigma);
}
```

```r
#hierachical conjuagate prior
data_list <- list(
  N = dim(X_design)[1],
  K = dim(X_design)[2],
  x = X_design,
  y = y_vector
)

model_hier <- stan_model(file='./prior_M2_hier.stan')

# Create a data list for Stan
set.seed(4827493)


# Fit the model to the data
stan_fit_hier <- sampling(model_hier,
                     data = data_list,
                     chains = 4,
                     iter = 5000)

# Print a summary of the results
```

```r
print(stan_fit_hier)

# Plot the posterior distributions
g <- plot(stan_fit_hier,
     pars=c("beta", "sigma"))


ggsave("./plots/M2_conj_hier_prior_4827493.pdf", g, width = 8, height = 6)
```

```stan
data {
  int<lower=1> N;   // number of data items
  int<lower=1> K;   // number of predictors, contain intercept
  matrix[N, K] x;   // design matrix
  vector[N] y;      // outcome vector


}
parameters {
  //indicator of predictor
  //int<lower=0> z[K-1];

  //parameters use conjugate
  vector[K] beta;       // coefficients for predictors
  real<lower=0> sigma2;  //variance

  //hyperparameters use uniform
  //for beta m0, C0
  vector[K] mu_beta;
  corr_matrix[K] C0;

  //for sigma
  real<lower=0> v0;
  real<lower=0> s0;
}
transformed parameters {
  real<lower=0> sigma = sqrt(sigma2);
  //vector[K] beta_ind = beta .* append_row(1, z);
}
model {
  //prior of z
  //bernoulli dist
  //z ~ bernoulli(0.5);

 // hyperprior
 C0 ~ lkj_corr(1.0);

 v0 ~ cauchy(0,1);
 s0 ~ cauchy(0,1);

 //priot
 sigma2 ~ inv_gamma(v0, s0);
 beta ~ multi_normal(mu_beta, sigma2 * C0);

 // likelihood
 y ~ normal(x * beta, sigma);
}
```

```
#model selection with z

K <- ncol(X_train)
inits <- list(z = rep(0, K-1),
              beta = rep(0, K),
              sigma2temp = 1,
              mu_beta = rep(0, K),
              v0 = 1,
              s0 = 1)

data_list <- list(
  N = dim(X_train)[1],
  K = dim(X_train)[2],
  x = X_train,
  y = y_train,
  Ik = diag(K),
  C0 = diag(K)
)

model_file <- "./prior_M2_ind.txt"
model.fit <- jags.model(model_file,
                    data = data_list,
                    inits = inits,
                    n.chains = 4)

update(model.fit, n.iter = 2000)  # Burn-in
model.samples <- coda.samples(model.fit,
                        variable.names = c("z", "beta", "sigma2temp",
                                           "beta_ind"), n.iter = 4000)

print(summary(model.samples))

posterior_samples <- as.matrix(model.samples)

posterior_mean <- apply(posterior_samples, 2, mean)
posterior_ci <- apply(posterior_samples, 2, function(x) quantile(x, c(0.25, 0.75)))

library(ggplot2)

# Assuming df is a data frame with columns: Parameter, Mean, Lower_CI, Upper_CI
df <- data.frame(Parameter = names(posterior_mean),
                 Mean = posterior_mean,
                 Lower_CI = posterior_ci[1,],
                 Upper_CI = posterior_ci[2,])


df.beta_ind <- df[12:22, ]
df.beta_ind$Parameter <- factor(df.beta_ind$Parameter,
                      levels = paste0('beta_ind[',11:1,']'))  # Specify the desired order

beta.sigma <- ggplot(df.beta_ind[-c(2,3, 6,7, 9),], aes(x = Mean, y = Parameter)) +
```

```r
    geom_errorbarh(aes(xmin = Lower_CI, xmax = Upper_CI), height = 0.2, col='red')+
  geom_point() +
  labs(title = "Posterior Mean and CI", x = "Posterior Mean", y = "Parameters")+
    theme(plot.title = element_text(hjust = 0.5))+
   scale_x_continuous(limits = c(-250, 400),
                      breaks = seq(-250, 400, by = 50))

show(beta.sigma)


df.z <- df[24:33, ]
df.z$Parameter <- factor(df.z$Parameter,
                       levels = paste0('z[',10:1,']'))  # Specify the
z <- ggplot(df.z, aes(x = Mean, y = Parameter)) +
  geom_point() +
  labs(title = "Posterior Mean and CI", x = "Posterior Mean", y = "Parameters")+
    theme(plot.title = element_text(hjust = 0.5))+
  xlim(0,1)+# Center the title+
 geom_vline(xintercept = 0.5, color = "red", linetype = "dashed")  # Add a red vertical line

show(z)

ggsave("./plots/selection_beta_ind.pdf", beta.sigma,width = 8, height = 6)
ggsave("./plots/selection_z.pdf", z,width = 8, height = 6)
```

```
model {
  # Prior for z
  for (i in 1:(K-1)) {
    z[i] ~ dbern(0.5)
  }

  # Hyperprior

  v0 ~ dt(0, 1, 1)T(0,)
  s0 ~ dt(0, 1, 1)T(0,)
  mu_beta ~ dmnorm(rep(0, K), inverse(t(x) %*% x))

  # Prior
  sigma2temp ~ dgamma(v0/2, s0/2)

  beta ~ dmnorm(mu_beta,  sigma2temp*inverse(t(x) %*% x))

  # Transform parameters

  beta_ind[1] <- beta[1]
  for (i in 2:K) {
    beta_ind[i] <- beta[i] * z[i-1]
  }

  # Likelihood
  for (i in 1:N) {
    y[i] ~ dnorm(sum(x[i,] * beta_ind), sigma2temp)
  }
}
```