Fitting partitioned MVM model with Bayes interval estimates in R

Fitting partitioned MVM model with Bayes interval estimates in R

You will need the haven package to import the partitioned data matrix from SAS and the rstan package to fit the model

```
#read packages necessary to run this analysis
library(haven)
library(rstan)
```

The first step is to import the partitioned data matrix that you obtain from running the %partitionedDataMatrix macro in SAS. This macro can be found in the following link

https://github.com/ElsaVazquez29/PartitionedDataMatrix/blob/master/partitioneddatamatrix%20macro.sas

```
library(haven)
#importing add health dataset with valid moment conditions for time-dependent
covariates obtained by running the %partitionedDataMatrix
library(haven)
add_health_valid <-
read_sas("C:/Users/vazquezarreolea/Desktop/disertation/obesity_valid.sas7bdat
",NULL)
```

Once you have imported the data, you will run a logistic regression model in Stan with the outcome variable, your time-independent covariates, and the lagged values of the time-dependent covariates with valid moment conditions

```
#the first thing you need to get is the total number of observations in your
dataset (N=total number of subjects*T), where T is the number of time periods
N=nrow(add_health_valid)

#define you binary outcome as y
y=add_health_valid$BMI

#declare your design matrix (X)
X=matrix(nrow=N, ncol=13, NA)



#filling our and declaring X matrix
X[,1]=rep(1, N) #the first component of your design matrix is the intercept
X[,2]=add_health_valid$RACE_ #race is a time-independent covariate
#the time-dependent covariates were depression (FEELINSCALE), TV hours
(TVHRS), physical activity (ACTIVITYSCALE), and alcohol (ALCOHOL)
#all four time-dependent covariates have valid moment conditions for cross-
```

```
sectional association
X[,3]=add_health_valid$FEELINGSCALE_0
X[,4]=add_health_valid$TVHRS_0
X[,5]=add_health_valid$ACTIVITYSCALE_0
X[,6]=add_health_valid$ALCOHOL_0
#all four time-dependent covariates have valid moment conditions for the lag-
1 association
X[,7]=add_health_valid$FEELINGSCALE_1
X[,8]=add_health_valid$TVHRS_1
X[,9]=add_health_valid$ACTIVITYSCALE_1
X[,10]=add_health_valid$ALCOHOL_1
#only physical activity and alcohol had valid moment conditions for the lag-2
association
X[,11]=add_health_valid$ACTIVITYSCALE_2
X[,12]=add_health_valid$ALCOHOL_2
#only physical activity had valid moment conditionsf or the lag-3 association
X[,13]=add_health_valid$ACTIVITYSCALE_3


#to fit the model in Stan you need the number of parameters to be estimated,
which is the number of columns in X (the design matrix)
P=ncol(X)
```

The following function is what fits the model in stan. In the function below, you only need to modify the code that refers to defining the priors, everything else is left as it is

```
#the following options have to be specified before running the model in stan
rstan_options(auto_write=TRUE)
options(mc.cores=parallel::detectCores())

#the following function in stan is the one that fits the model, you can
modify it to fit your model depending on your data

partitioned_bayes <- "
data {
// In this section, we define the data that must be passed to Stan (from
whichever environment you are using)
int N; // number of observations
int P; // number of covariates
matrix[N, P] X; //covariate matrix
int y[N]; //outcome vector
}
parameters {
// Define the parameters that we will estimate, as well as any restrictions
on the parameter values (standard deviations can't be negative...)
vector[P] beta; // the regression coefficients
}

transformed parameters  {
// Probability trasformation from linear predictor
```

```
real<lower=0> odds[N];  //the odds of an event cannot be negative
real<lower=0, upper=1> prob[N];  //probability of an event is always between
0 and 1

for (i in 1:N) {
odds[i] =exp(X[i,]*beta);
prob[i] = odds[i] / (odds[i] + 1);
}
}

model {
//Define the priors (here we have informative priors)
beta[1]~normal(0,10000); //prior for the intercept
beta[2] ~ normal(-0.113, 10000); //prior for race
beta[3] ~ normal(0.255,10000); //prior for cross-sectional association of
depression
beta[4] ~ normal(-0.073, 10000); //prior for cross-sectional association of
tv hours
beta[5] ~ normal(0.457, 10000); //prior for cross-sectional assocation of
physical activity
beta[6] ~ normal(0, 10000); // prior for cross-sectional assocation of
alcohol
beta[7] ~ normal(0.255, 10000); // prior for lag-1 assocciation of depression
beta[8] ~ normal(-0.073,10000); //prior for lag-1 assocation of tv hours
beta[9] ~ normal(0.457, 10000); //prior for lag-1 assocation of physical
activity
beta[10] ~ normal(0, 10000); //prior for lag-1 association of alcohol
beta[11] ~ normal(-0.073, 10000); //prior for lag-2 association of physical
activity
beta[12] ~ normal(0, 10000); //prior for lag-2 association of alcohol
beta[13] ~ normal(-0.073, 10000); // prior for lag-3 association of physical
activity



y~bernoulli(prob);
}
"
#here you compile the function
compiled_function=stan_model(model_code =partitioned_bayes )

# And make the function available to the user in R
expose_stan_functions(compiled_function)

#the following list contains the design matrix X, the total number of
observations, the outcome variable and the number of parameters
data_list_2 <- list(X = X, N = N, y = y, P = P)


#this is the part the fits the model, you are using the partitioned_bayes
```

```
function, the data in data_list_2, and are specifying that you want to run 3
chains with 2000 iterations each using 3 cores
correct_fit <- stan(model_code = partitioned_bayes, data = data_list_2, cores
= 3,  chains =3 , iter = 20)

#the following command prints the results of the fitted model
print(correct_fit, pars = c("beta"))
```