

MCAR 30% missing - DPMPM

```
# sample MCAR dataset from PUMS
source("../utils/sampleMCAR.R")
n = 10000
missing_col = c(1,3,7,9,10,11)
missing_prob = 0.3
set.seed(0)

output_list <- sampleMCAR(n, missing_prob)
df <- output_list[['df']]
df_observed <- output_list[['df_observed']]
```

DPMPM

Multiple imputation using NPBayesImputeCat package

Ref: <https://cran.r-project.org/web/packages/NPBayesImputeCat/NPBayesImputeCat.pdf>

1. Create and initialize the Rcpp_Lcm model object using CreateModel with the following arguments:
 - X: dataframe to be imputed = df
 - MCZ: dataframe with the definition of structural zero = NULL
 - K: the maximum number of mixture components = 40
 - Nmax: An upper truncation limit for the augmented sample size = 0
 - aalpha: the hyper parameter alpha in stick-breaking prior = 0.25
 - balpha: the hyper parameter beta in stick-breaking prior = 0.25
 - seed = 0
2. Set the tracer for the sampling process
 - k_star: the effective cluster number
 - psi: conditional multinomial probabilities
 - ImputedX: imputation result
3. Run the model using the method Run of Rcpp_Lcm class with the following arguments:
 - burnin = 10000
 - iter = 10000
 - thinning = 5
4. Obtain result

```
N = 40
Mon = 10000
B = 10000
thin.int = 5

# 1. Create and initialize the Rcpp_Lcm model object
model = CreateModel(X = df_observed, MCZ = NULL, K = N, Nmax = 0,
                     aalpha = 0.25, balpha = 0.25, seed = 0)
# 2. Set tracer
model$SetTrace(c('k_star', 'psi', 'ImputedX', 'alpha'), Mon)

# 3. Run model using Run(burnin, iter, thinning)
model$Run(B, Mon, thin.int)
```

```

# Extract results and format output
output <- model$GetTrace()
k_star <- output$k_star
psi <- output$psi
imputed_df <- output$ImputedX
alpha <- output$alpha

#retrieve parameters from the final iteration
result <- model$snapshot

#convert ImputedX matrix to dataframe, using proper factors/names etc.
ImputedX <- GetDataFrame(result$ImputedX,df)

# extract 5 imputed dataset from DP model
imputation_index = as.integer(seq(1,dim(imputed_df)[1], length.out = 5))
imputation_list = list()
levels = c(7,7,7,19,5,4,7,2,17,3,13)
for (i in 1:length(imputation_index)) {
  index = imputation_index[i]
  # need to plus 1 here because the class index of DP function starts at 0
  d = imputed_df[index,] + 1
  dim(d) = dim(t(df_observed))
  d = data.frame(t(d))
  colnames(d) = colnames(df_observed)
  # format columns of d
  for (col_index in 1:ncol(df_observed)) {
    d[,col_index] = factor(d[,col_index], levels = 1:levels[col_index], ordered = TRUE)
  }
  imputation_list[[i]] = d
}

```

Diagnostics

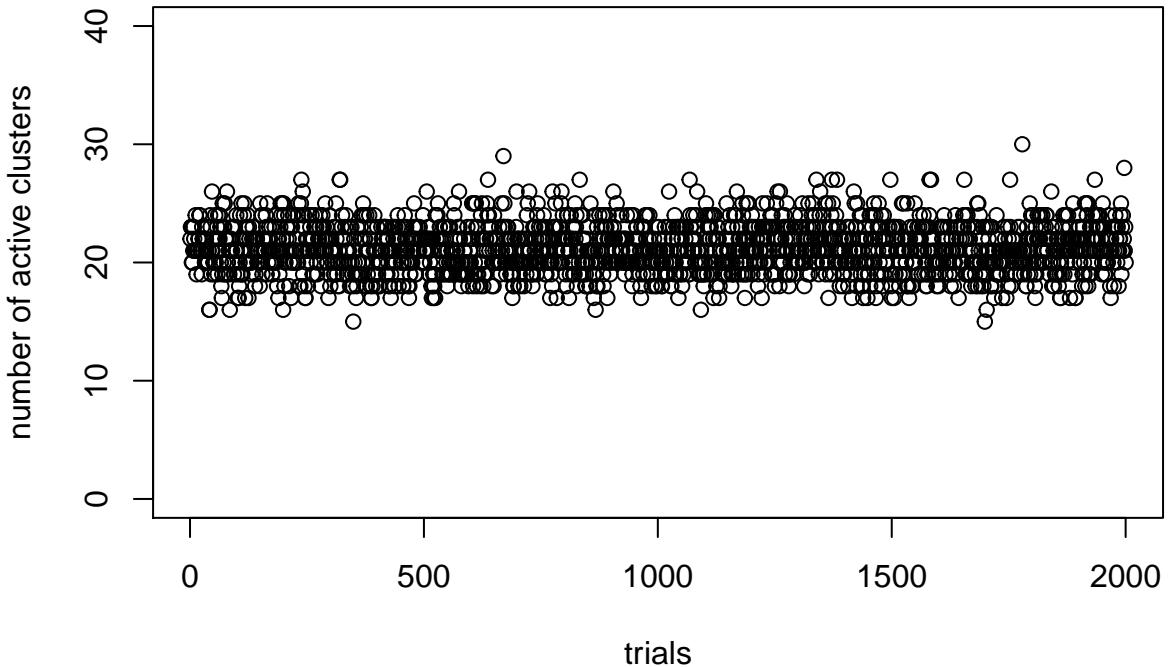
```

source("../utils/create_report.R")
create_report(imputation_list, max_nway=4, missing_col, df_observed)

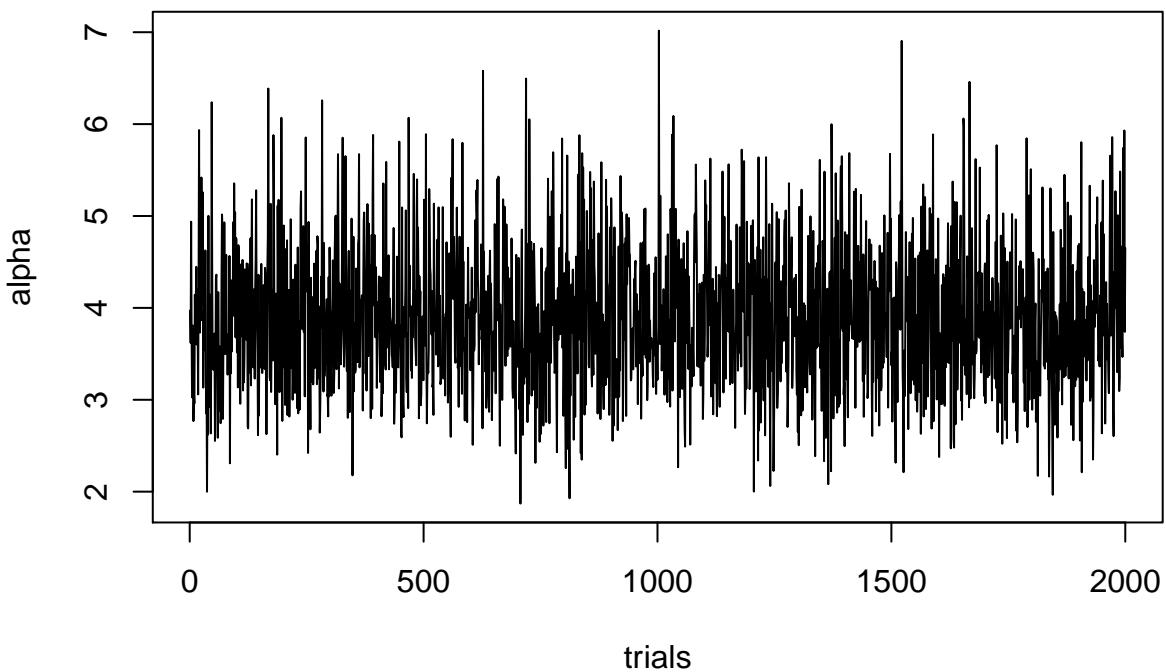
## ##### Coverage #####
## Coverage 1 way: 98.15 percent
## Coverage 2 way: 96.2 percent
## Coverage 3 way: 97.96 percent
## Coverage 4 way: 98.83 percent
##
## ##### RMSE #####
## RMSE 1 way: 0.00267
## RMSE 2 way: 0.00129
## RMSE 3 way: 0.000488
## RMSE 4 way: 0.000173
##
## ##### MAE #####
## MAE 1 way: 0.001985
## MAE 2 way: 0.000713
## MAE 3 way: 0.000199
## MAE 4 way: 5.2e-05

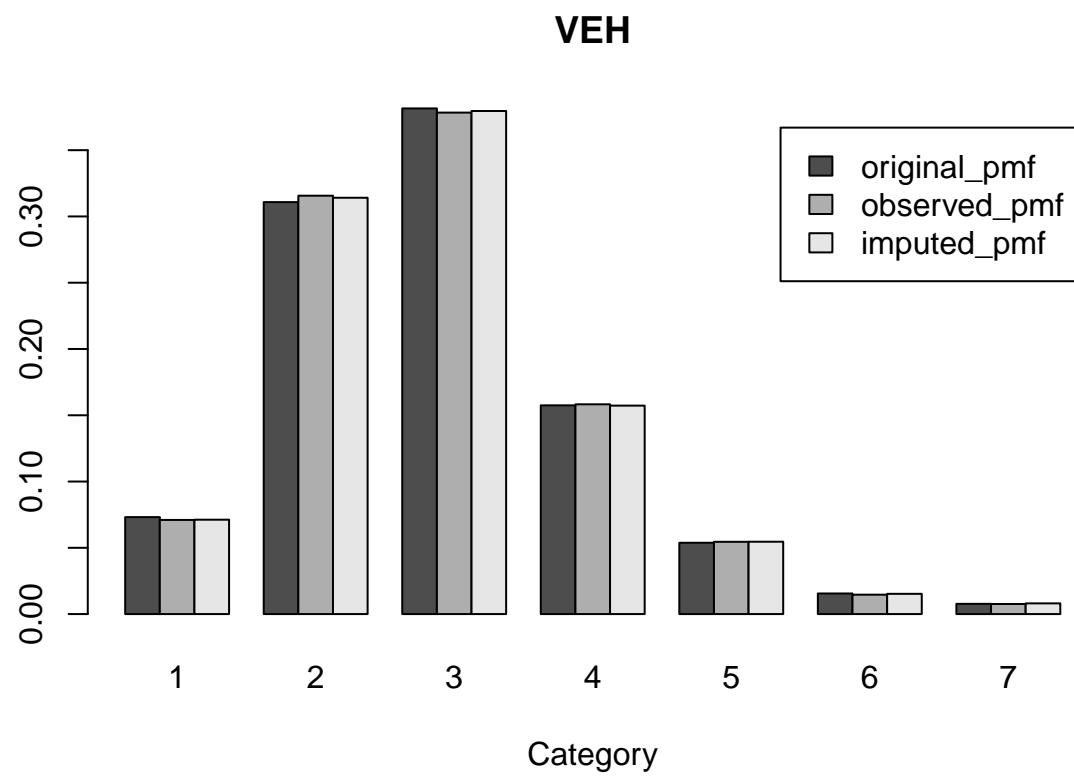
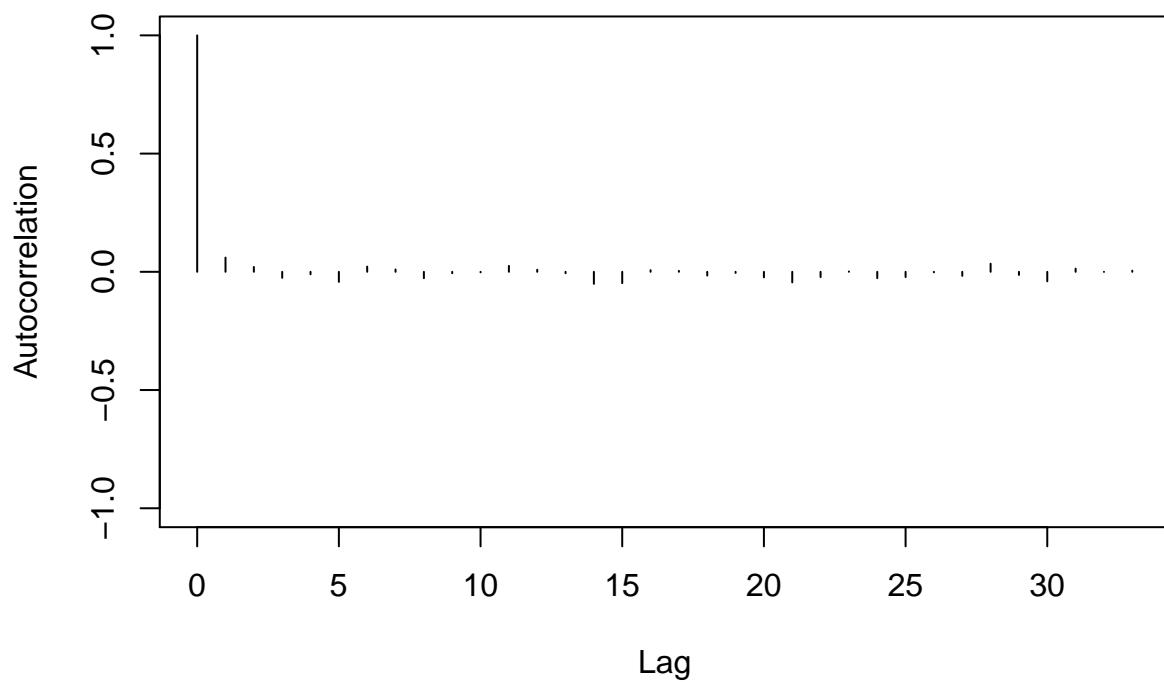
```

Number of clusters used over time

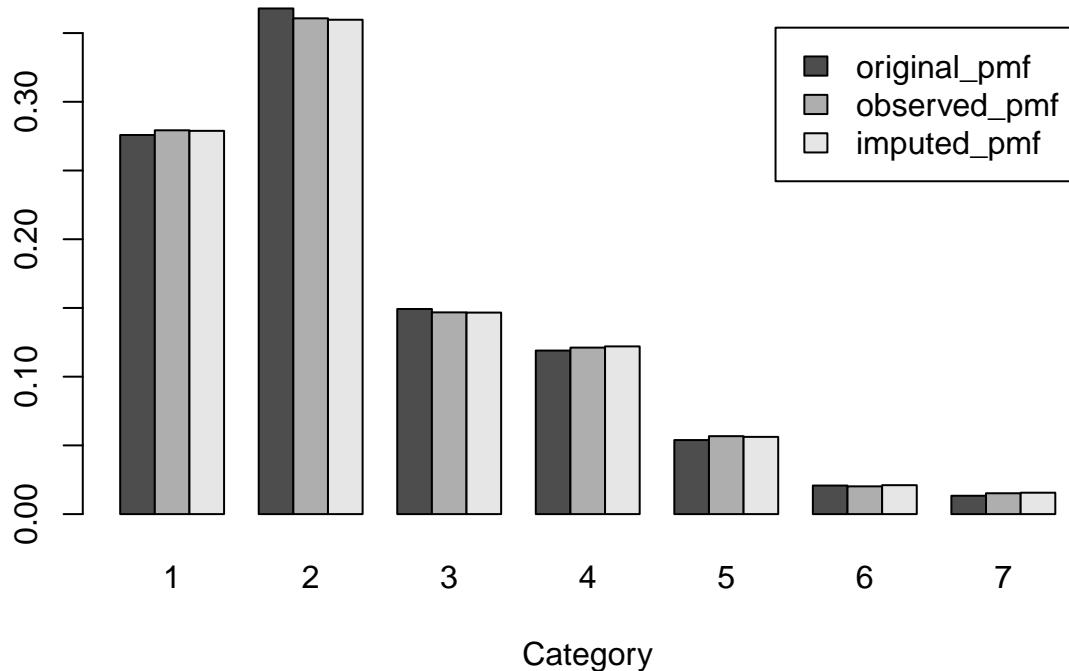


alpha value for the stick breaking process

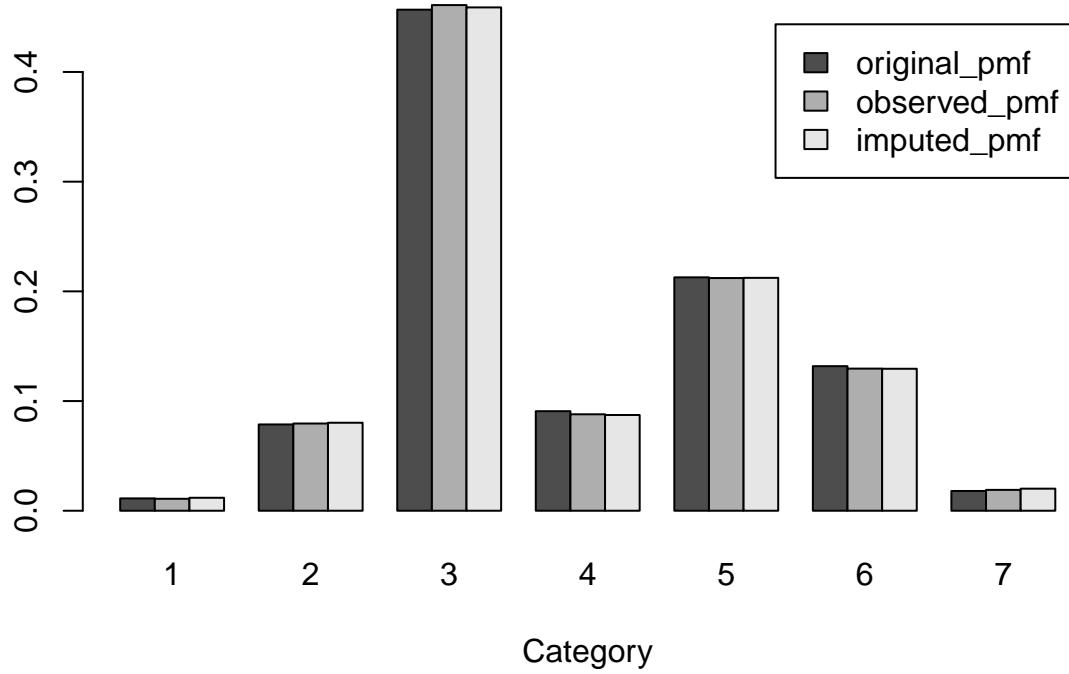




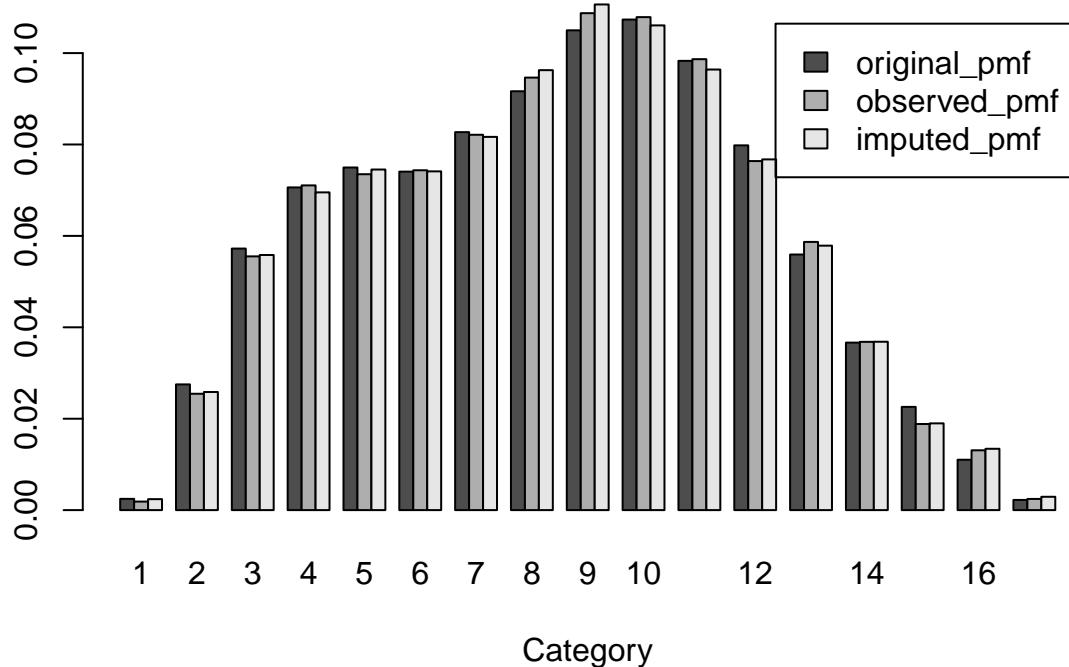
NP



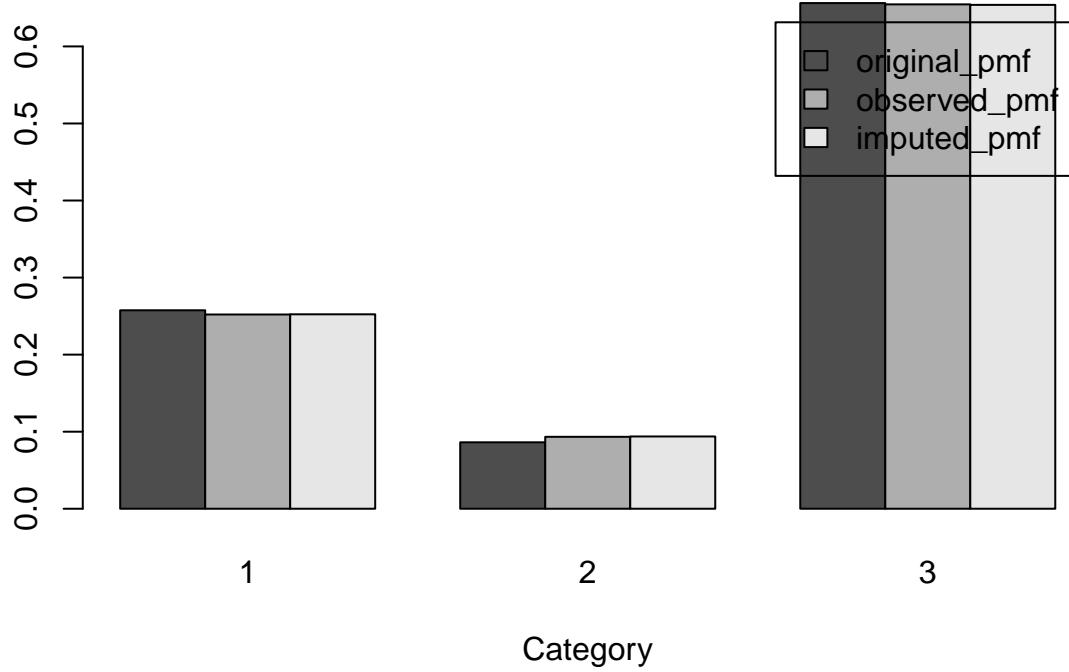
SCHL



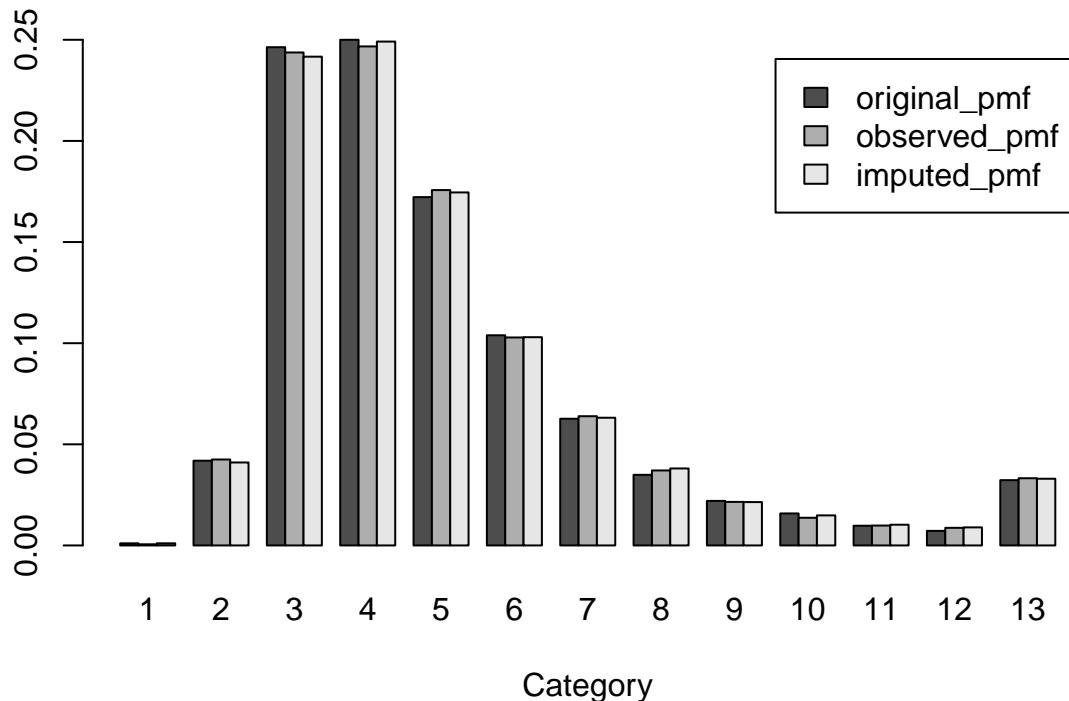
AGEP



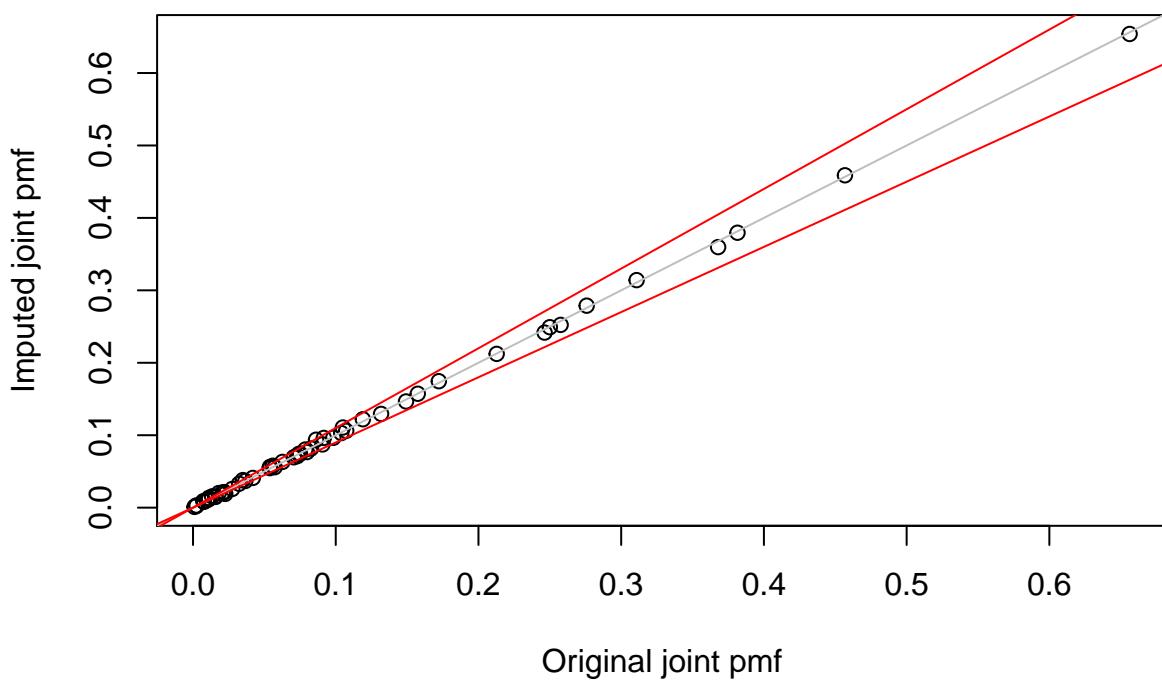
WKL



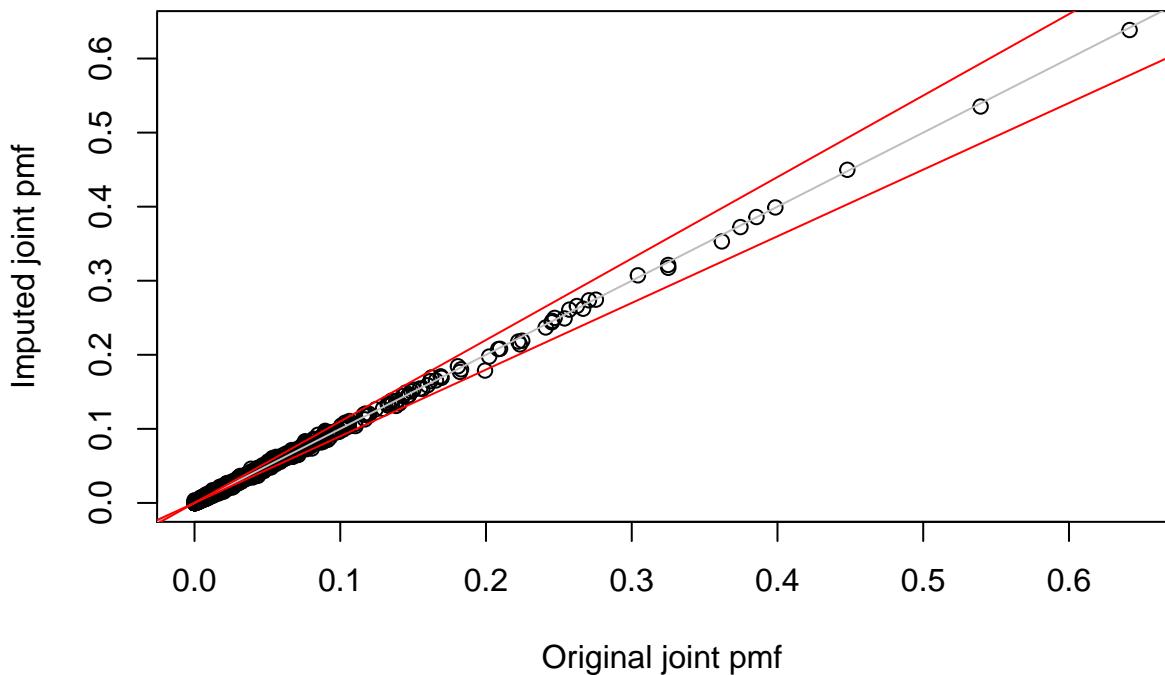
PINCP



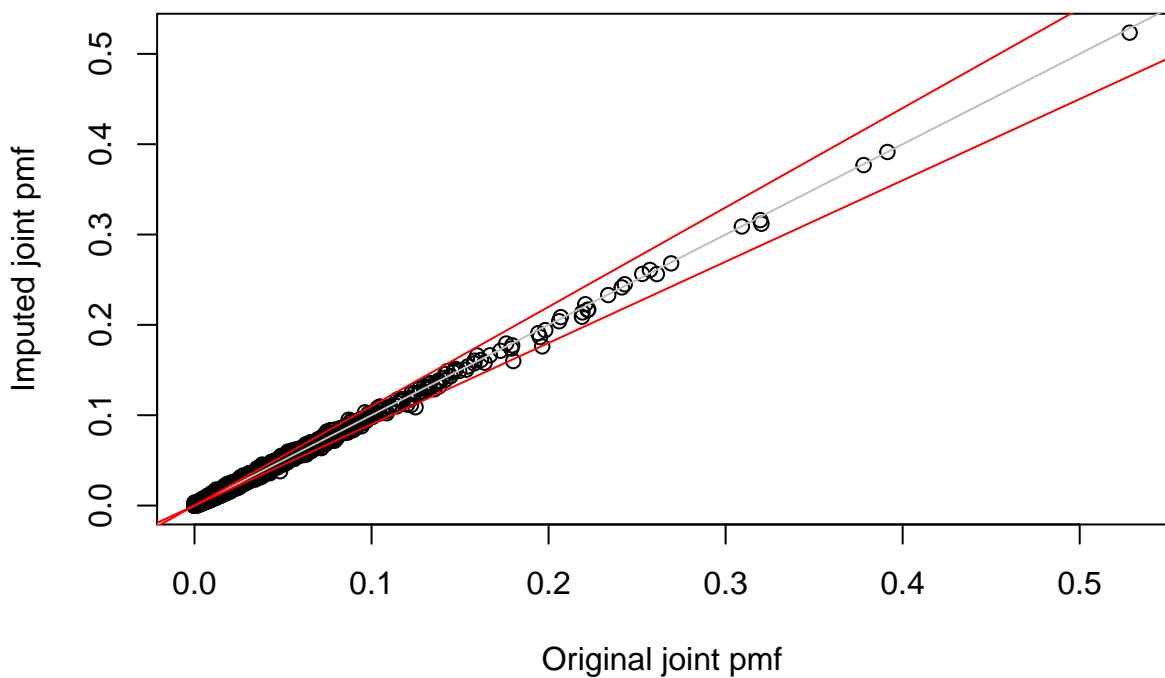
Assess imputed pmf: 1 way



Assess imputed pmf: 2 way



Assess imputed pmf: 3 way



Assess imputed pmf: 4 way

