MCAR 30% missing - DPMPM

```r
# 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 imptuted = 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 probabilties
- 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

```r
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)
```

```r
# Extract results
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)
```

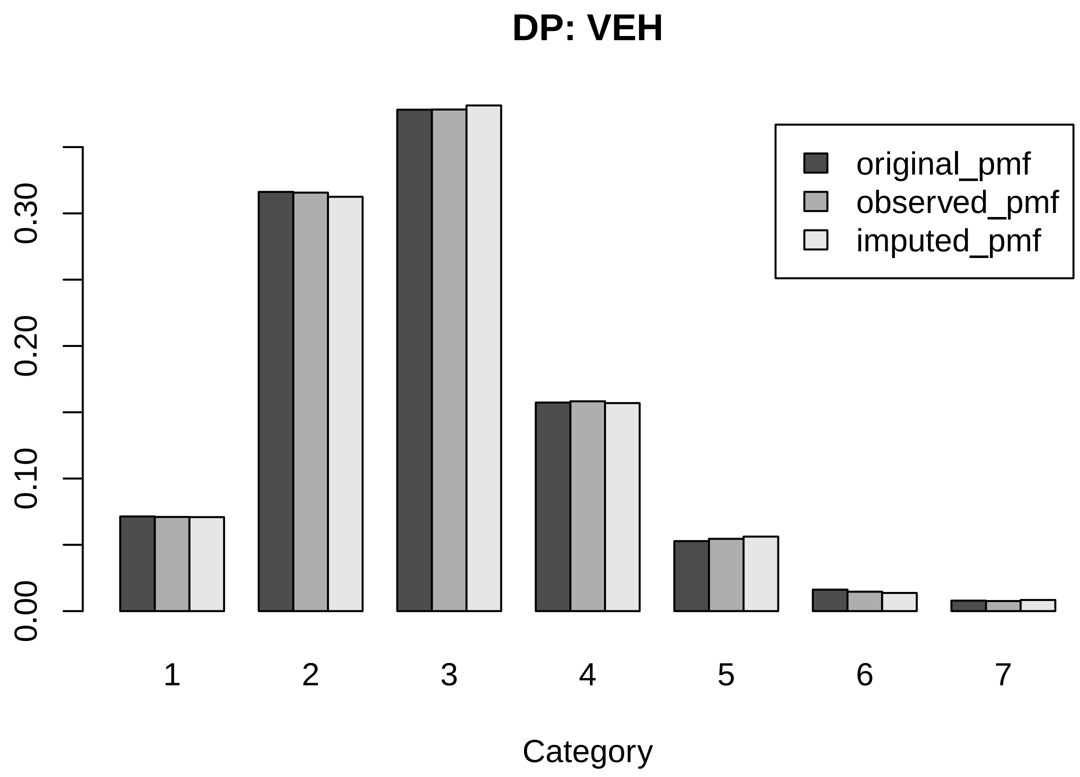
```r
# extract 5 imputed dataset from DP model
imputation_index = as.integer(seq(1,dim(imputed_df)[1], length.out = 5))
imputation_list = list()
levels = c(7,7,7,19,5,4,7,2,17,3,13)
for (index in imputation_index) {
  # need to plus 1 here because the class index of DP function starts at 0
  d = imputed_df[index,] + 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[[index]] = d
}
```

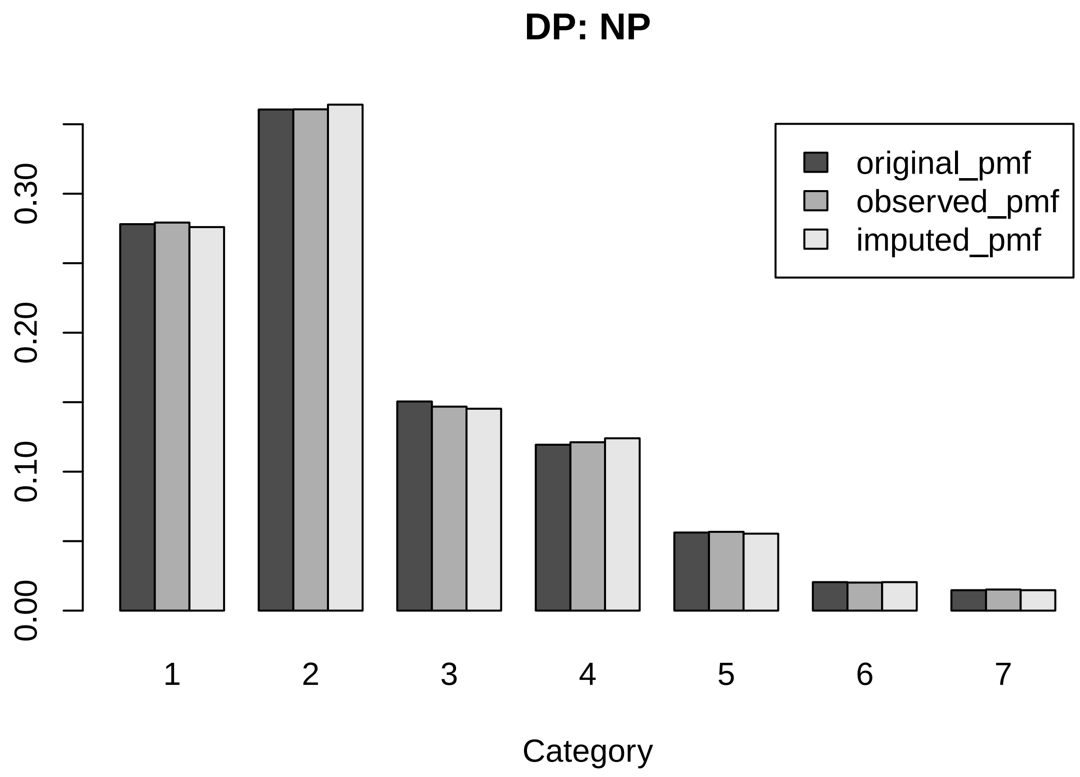
Diagnostics

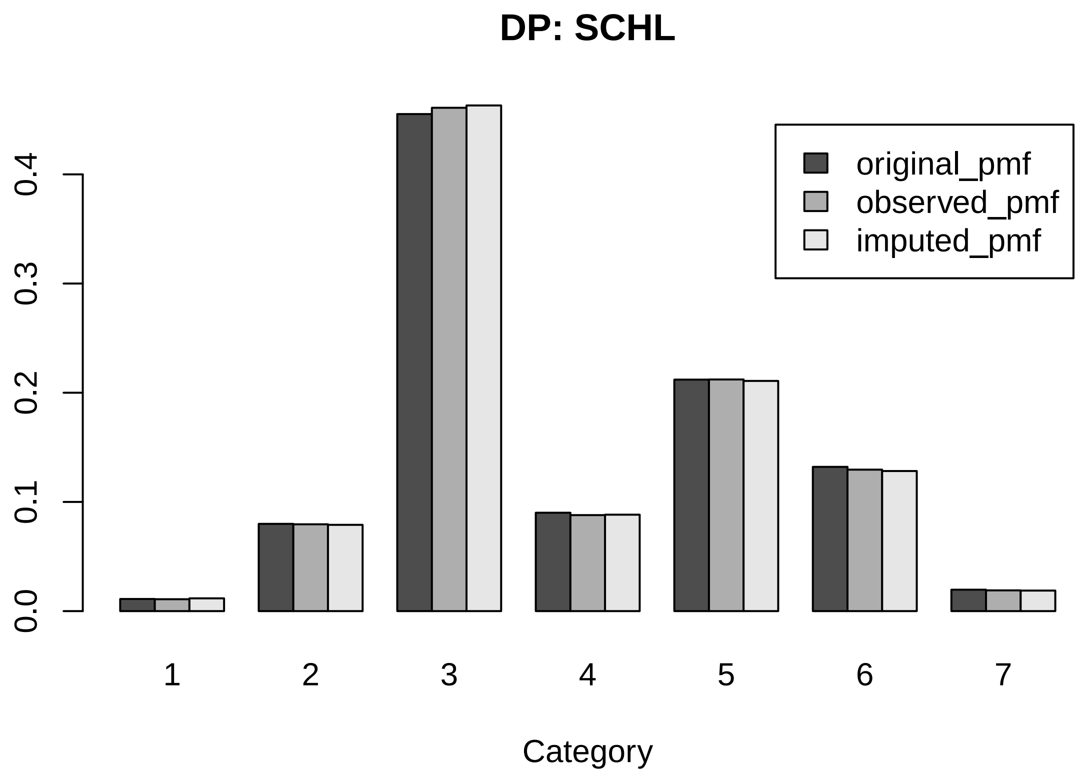
```r
d1 = imputation_list[[1]]
d2 = imputation_list[[2]]
d3 = imputation_list[[3]]
d4 = imputation_list[[4]]
d5 = imputation_list[[5]]
imputed_sets = rbind(d1, d2, d3, d4, d5)
```

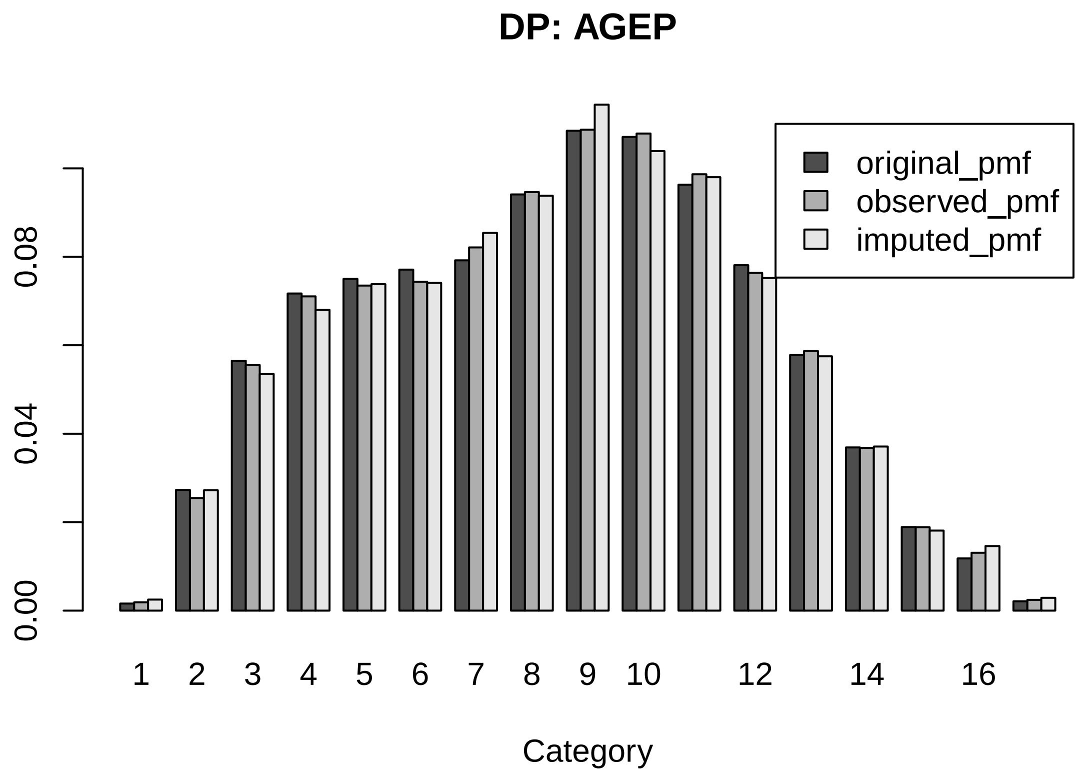
Assess bivariate joint distribution

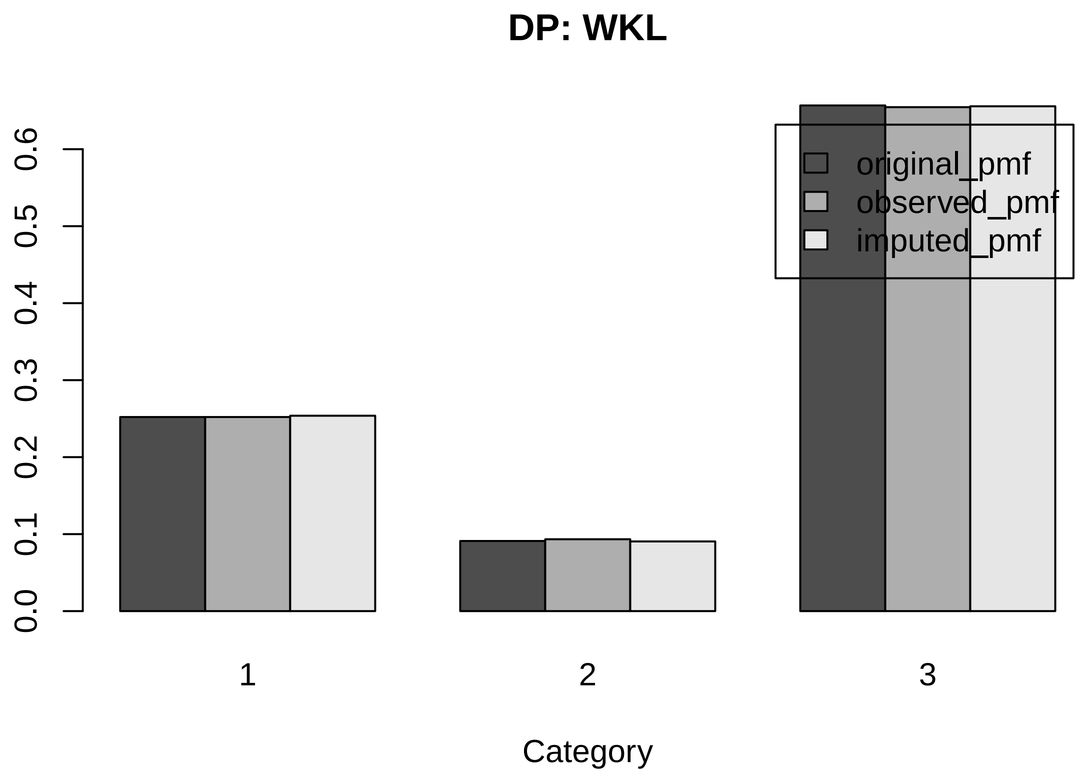
Assess trivariate joint distribution

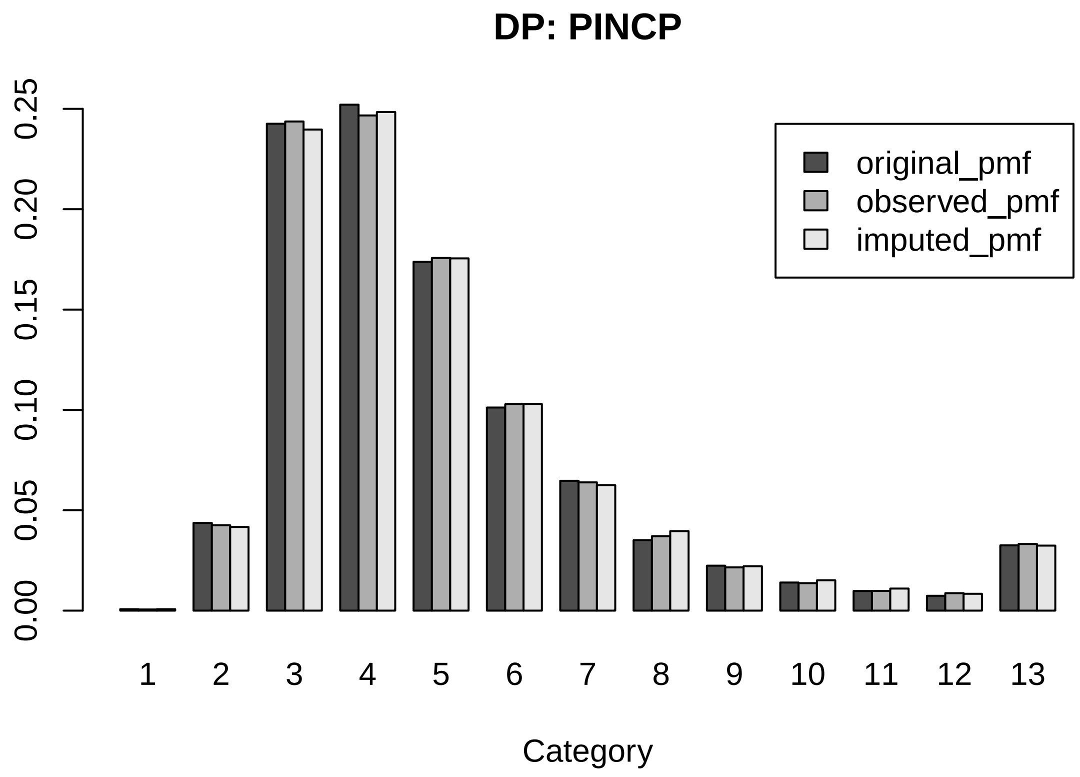

DP: VEH
original_pmf
observed_pmf
imputed_pmf
0.00
0.10
0.20
0.30
1
2
3
4
5
6
7
Category


DP: NP
original_pmf
observed_pmf
imputed_pmf
0.00
0.10
0.20
0.30
1
2
3
4
5
6
7
Category

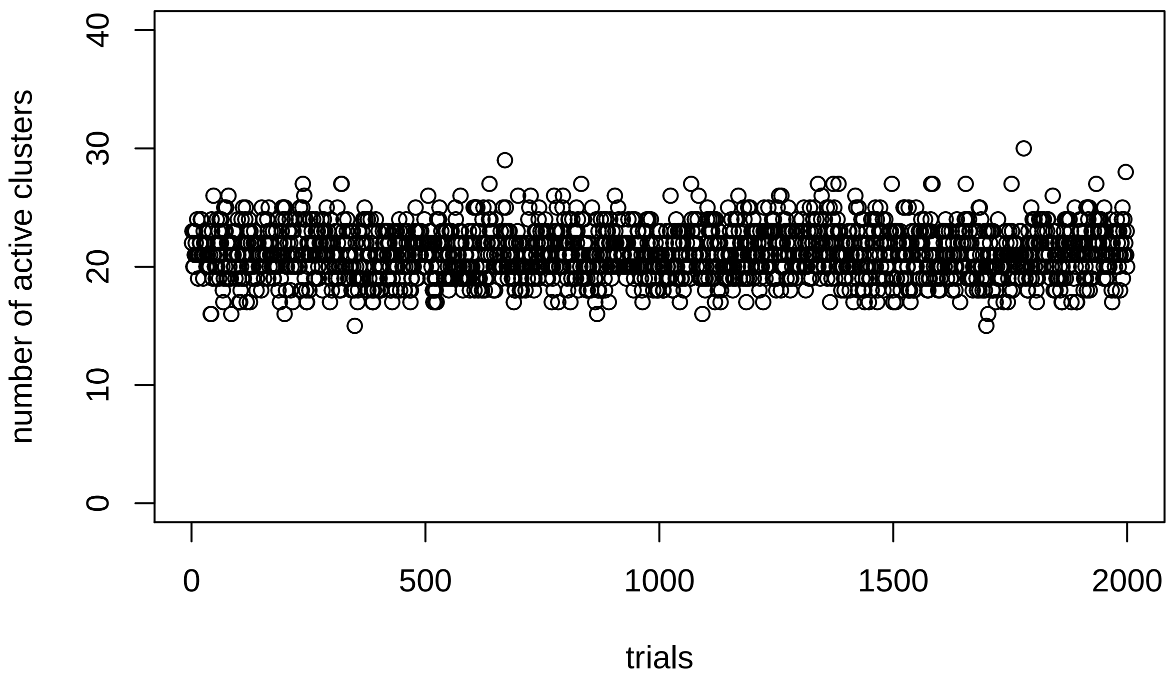

DP: SCHL
original_pmf
observed_pmf
imputed_pmf
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
Category


DP: AGEP
original_pmf
observed_pmf
imputed_pmf
0.00
0.04
0.08
1
2
3
4
5
6
7
8
9
10
12
14
16
Category

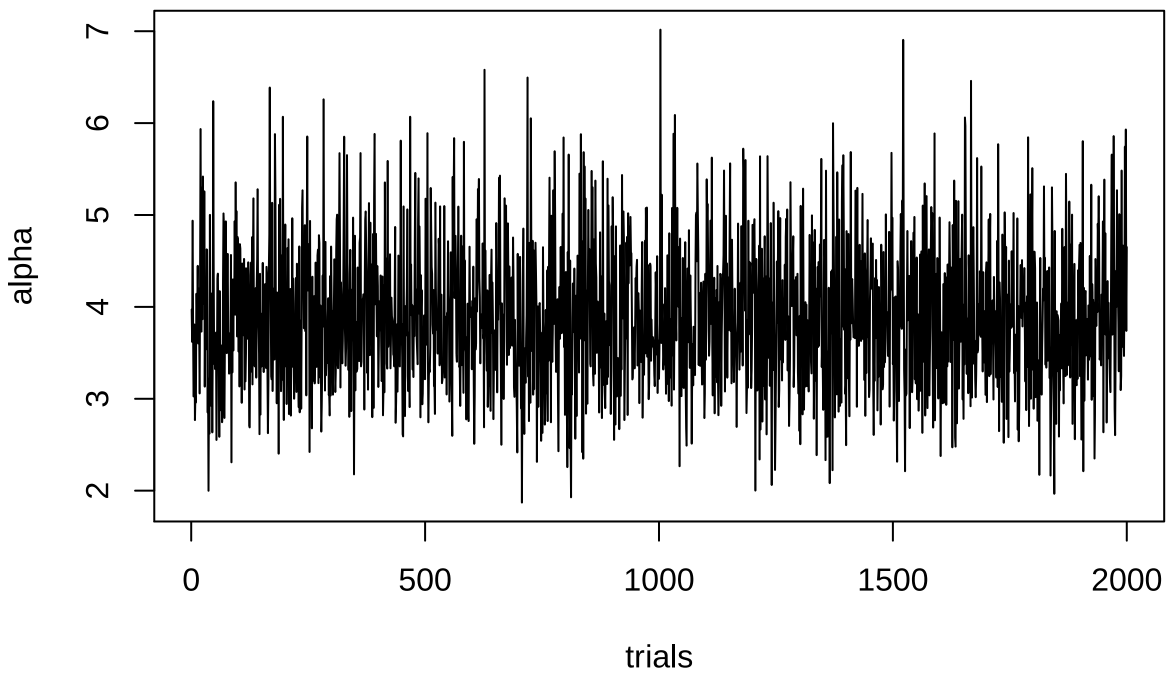

DP: WKL
0.0
0.1
0.2
0.3
0.4
0.5
0.6
original_pmf
observed_pmf
imputed_pmf
1
2
3
Category

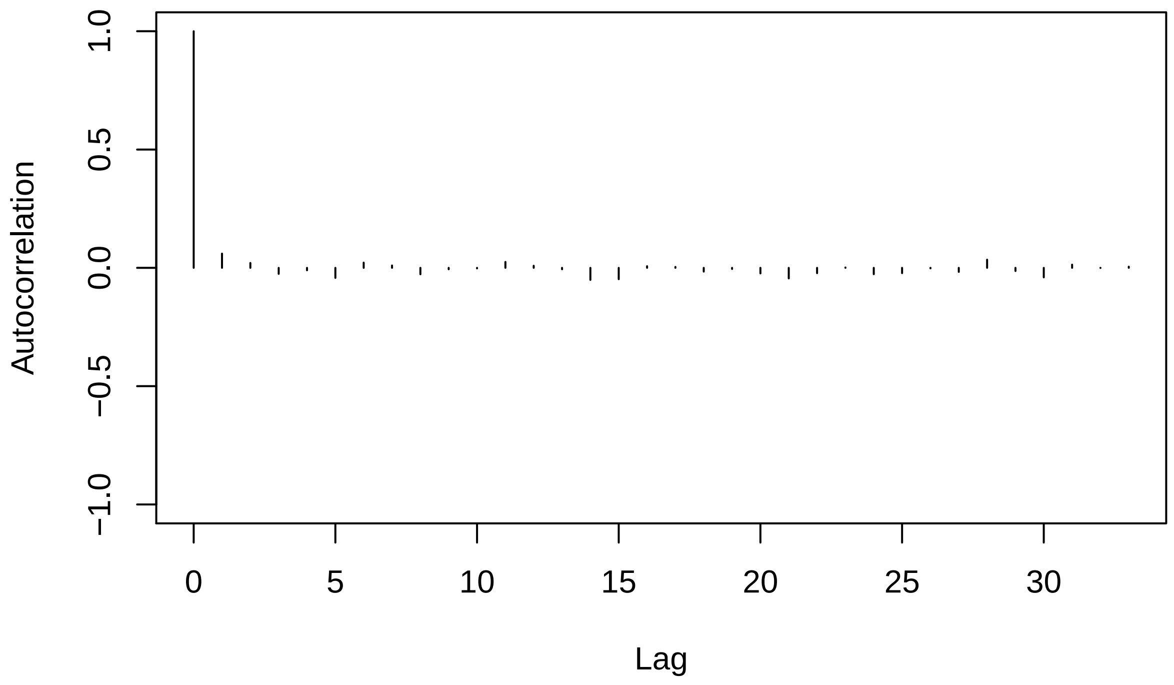

DP: PINCP
0.00
0.05
0.10
0.15
0.20
0.25
original_pmf
observed_pmf
imputed_pmf
1
2
3
4
5
6
7
8
9
10
11
12
13
Category

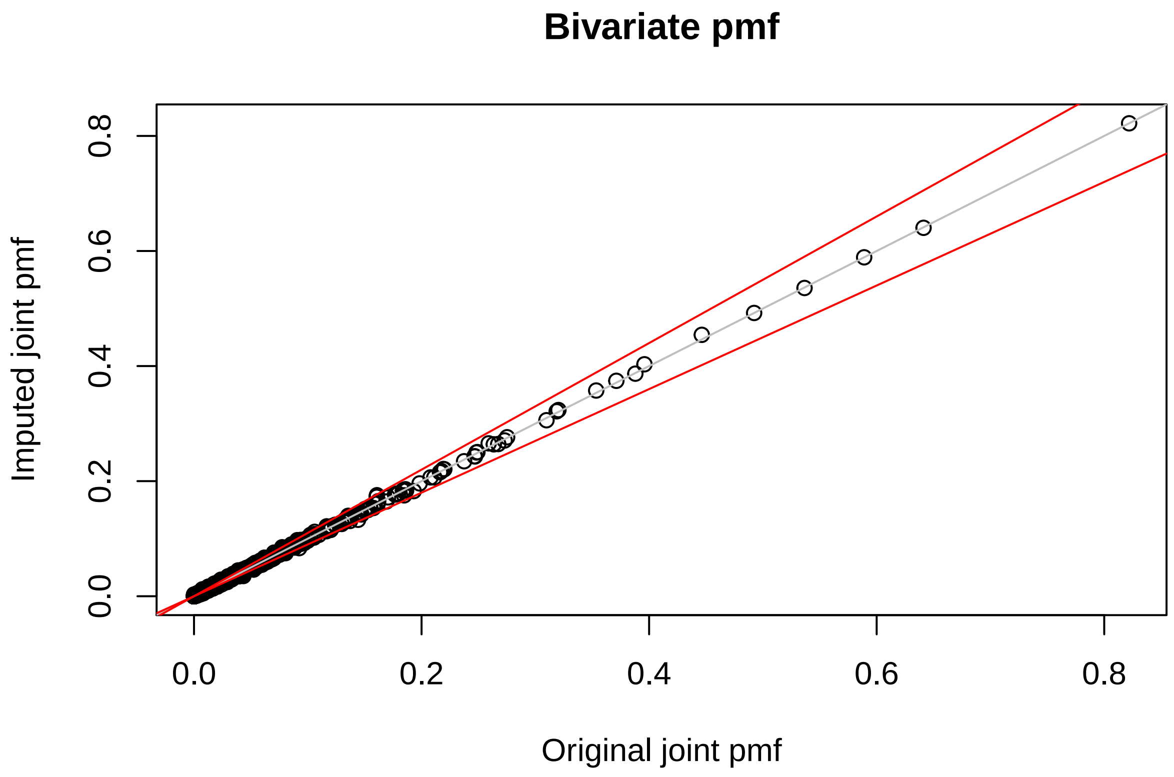
Number of clusters used over time

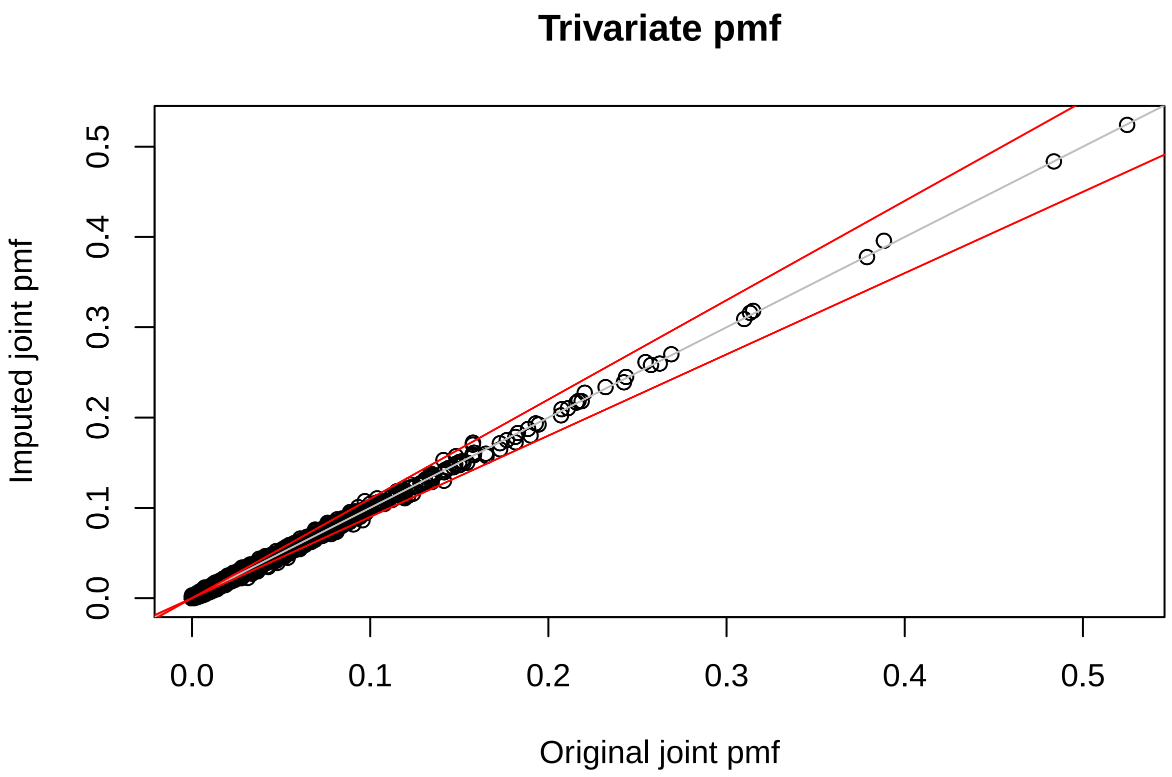


alpha value for the stick breaking process




Autocorrelation
Lag
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30


Bivariate pmf
Imputed joint pmf
Original joint pmf
0.0
0.2
0.4
0.6
0.8


Trivariate pmf
Imputed joint pmf
Original joint pmf
0.0
0.1
0.2
0.3
0.4
0.5