Testing different imputation methods on PUMS (MCAR) - DPMPM

```r
# load dataset: df
load('../Datasets/ordinalPUMS.Rdata')

# take 10,000 samples: df
set.seed(0)
n = 10000
sample <- sample(nrow(df), size = 10000)
df <- df[sample,]

# create MCAR scneario with 30% chance of missing: df_observed
missing_prob = 0.3
df_observed <- df
missing_col = colnames(df)[c(1,3,5,7,9,11)]
for (col in missing_col) {
  missing_ind <- rbernoulli(n,p = missing_prob)
  df_observed[missing_ind, col] <- NA
}
```

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 = 300
- iter = 2000
- thinning = 5

4. Obtain result

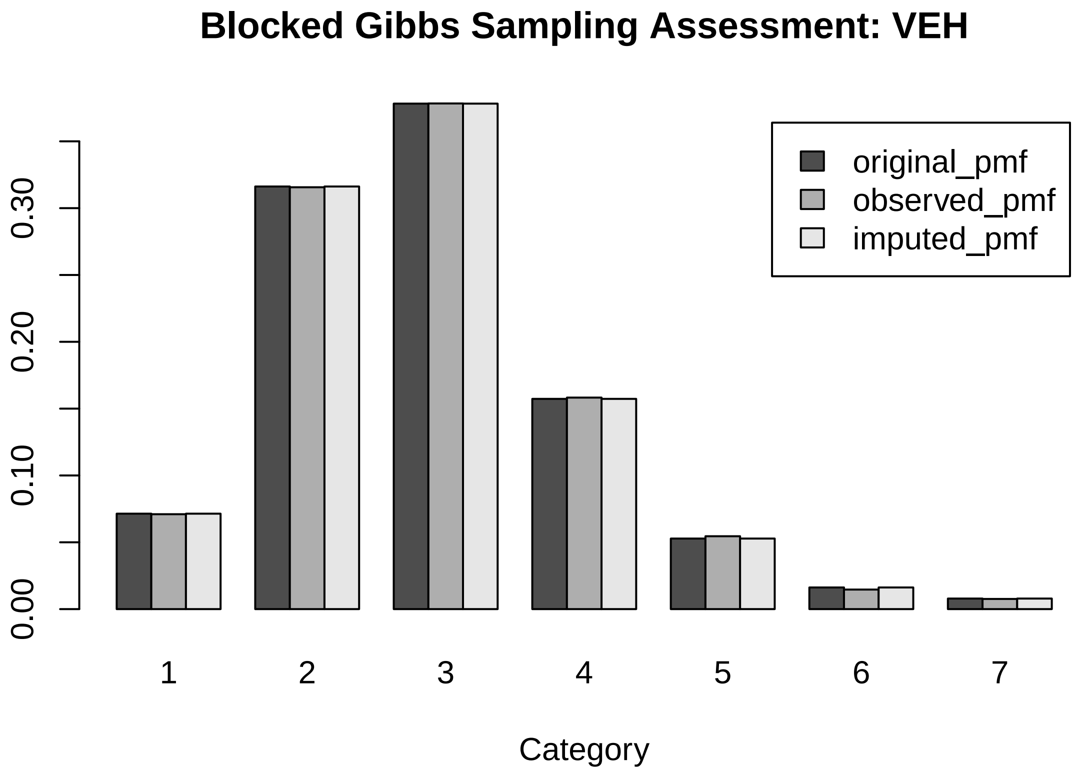
```r
N = 40
Mon = 2000
B = 300
thin.int = 5
```

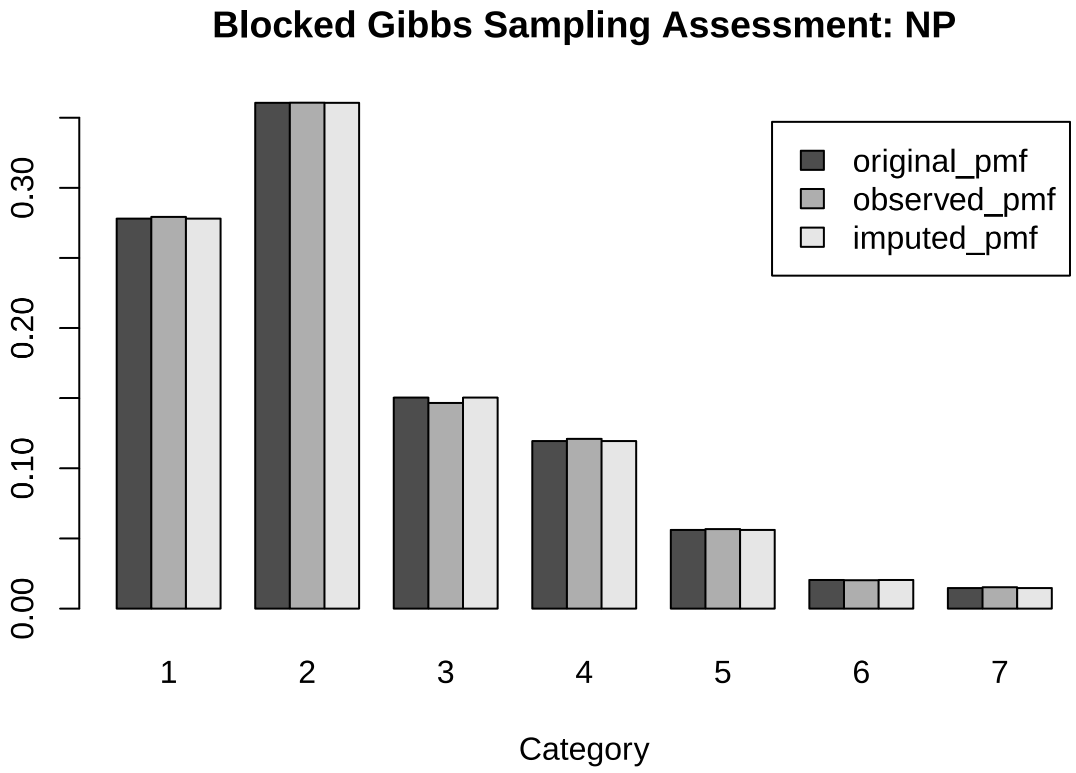
```r
# 1. Create and initialize the Rcpp_Lcm model object
model = CreateModel(X = df, MCZ = NULL, K = N, Nmax = 0,
                    aalpha = 0.25, balpha = 0.25, seed = 0)
# 2. Set tracer
model$SetTrace(c('k_star', 'psi', 'ImputedX', 'alpha'),Mon)

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

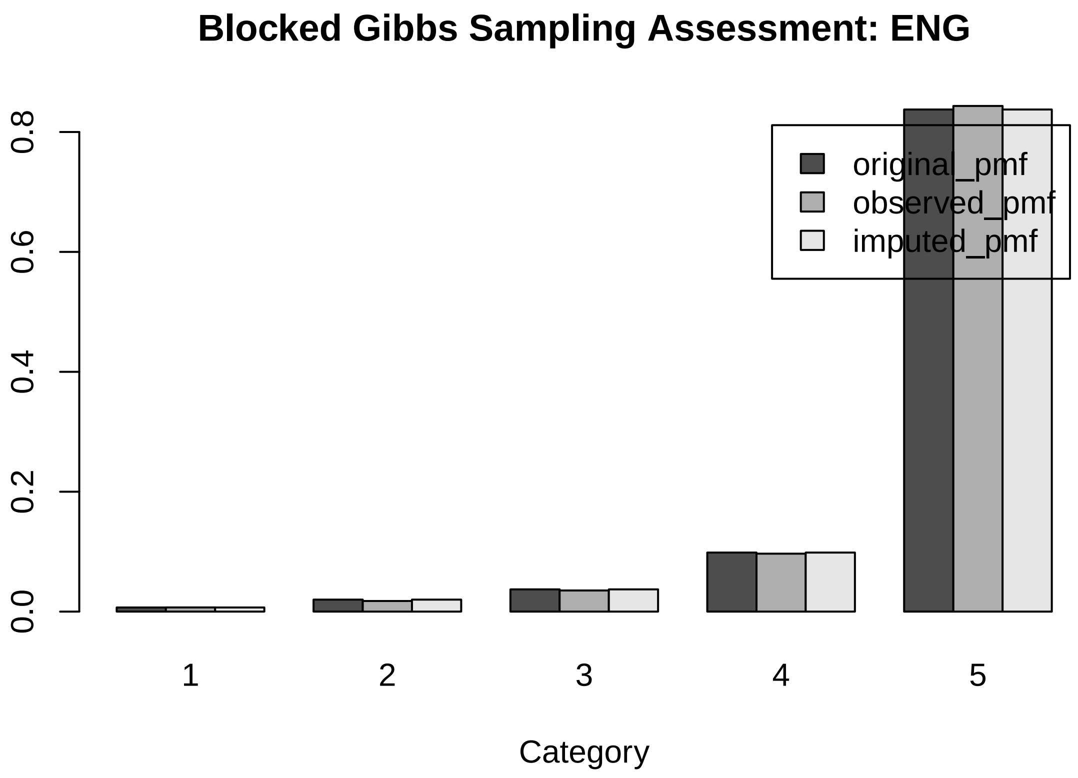
```r
# Extract results
output <- model$GetTrace()
k_star <- output$k_star
psi <- output$psi
imputed_df <- output$ImputedX
alpha <- output$alpha

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

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

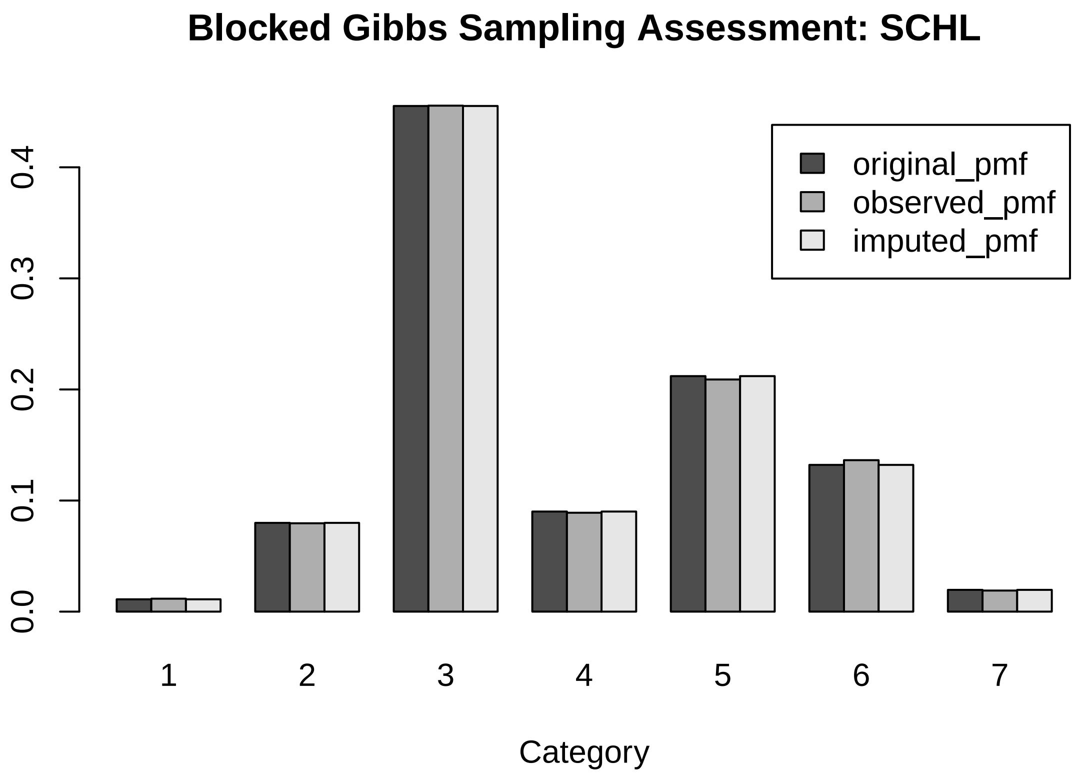
Diagnostics

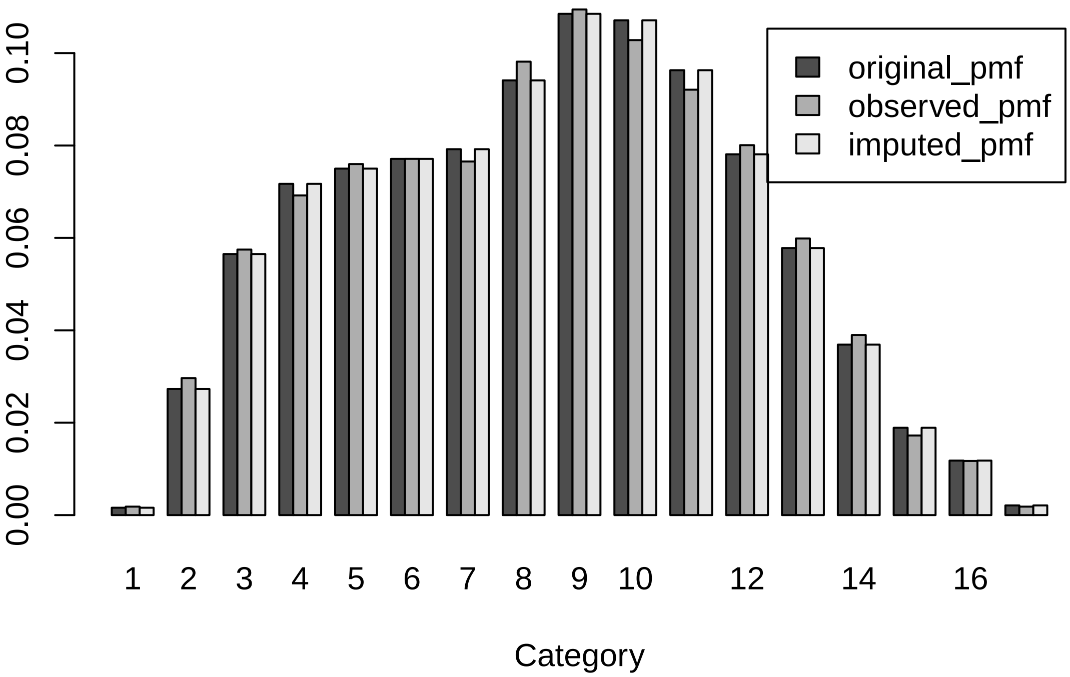
```r
for (var_index in c(1,3,5,7,9,11)) {
  y_original = df[,var_index]
  original_pmf = table(y_original)/length(y_original)

  # Observed distribution
  missing_indicator = is.na(df_observed)[,var_index]
  y_observed = y_original[!missing_indicator]
  observed_pmf = table(y_observed)/length(y_observed)

  # Extract variable from imputed data
  imputed_pmf = table(imputed_df[,seq(var_index, dim(imputed_df)[2], dim(df)[2])])
  imputed_pmf = imputed_pmf/sum(imputed_pmf)

  results = rbind(original_pmf,observed_pmf,imputed_pmf)
  colnames(results)<- 1:dim(imputed_pmf)
  barplot(results, xlab = 'Category', beside = TRUE,
          legend = TRUE,
          main = paste('Blocked Gibbs Sampling Assessment:', colnames(df)[var_index]))
}
```


Blocked Gibbs Sampling Assessment: VEH
0.00
0.10
0.20
0.30
1
2
3
4
5
6
7
Category
original_pmf
observed_pmf
imputed_pmf

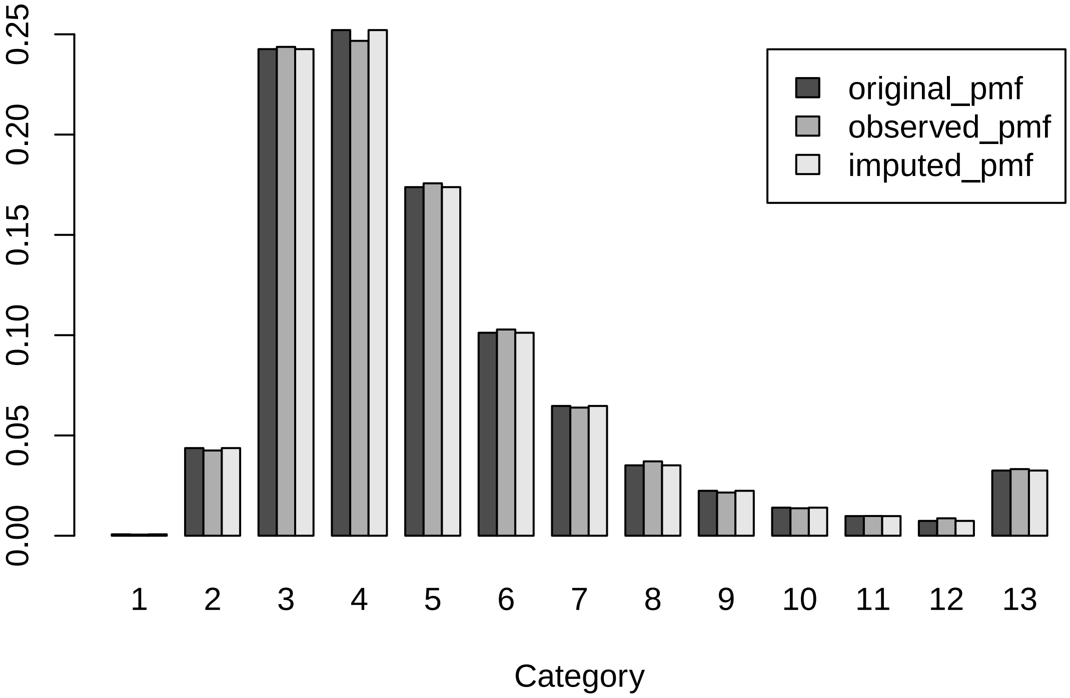

Blocked Gibbs Sampling Assessment: NP
0.00
0.10
0.20
0.30
1
2
3
4
5
6
7
Category
original_pmf
observed_pmf
imputed_pmf


Blocked Gibbs Sampling Assessment: ENG
0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
Category
original_pmf
observed_pmf
imputed_pmf

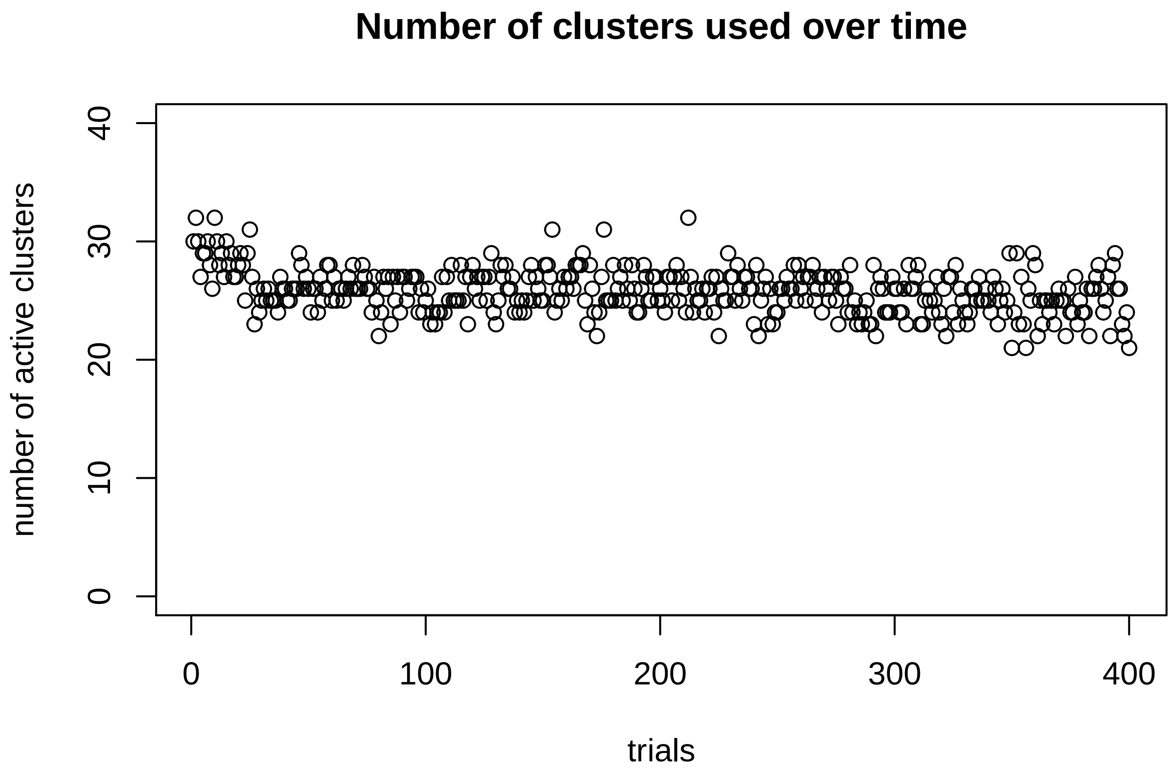

Blocked Gibbs Sampling Assessment: SCHL
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
Category
original_pmf
observed_pmf
imputed_pmf

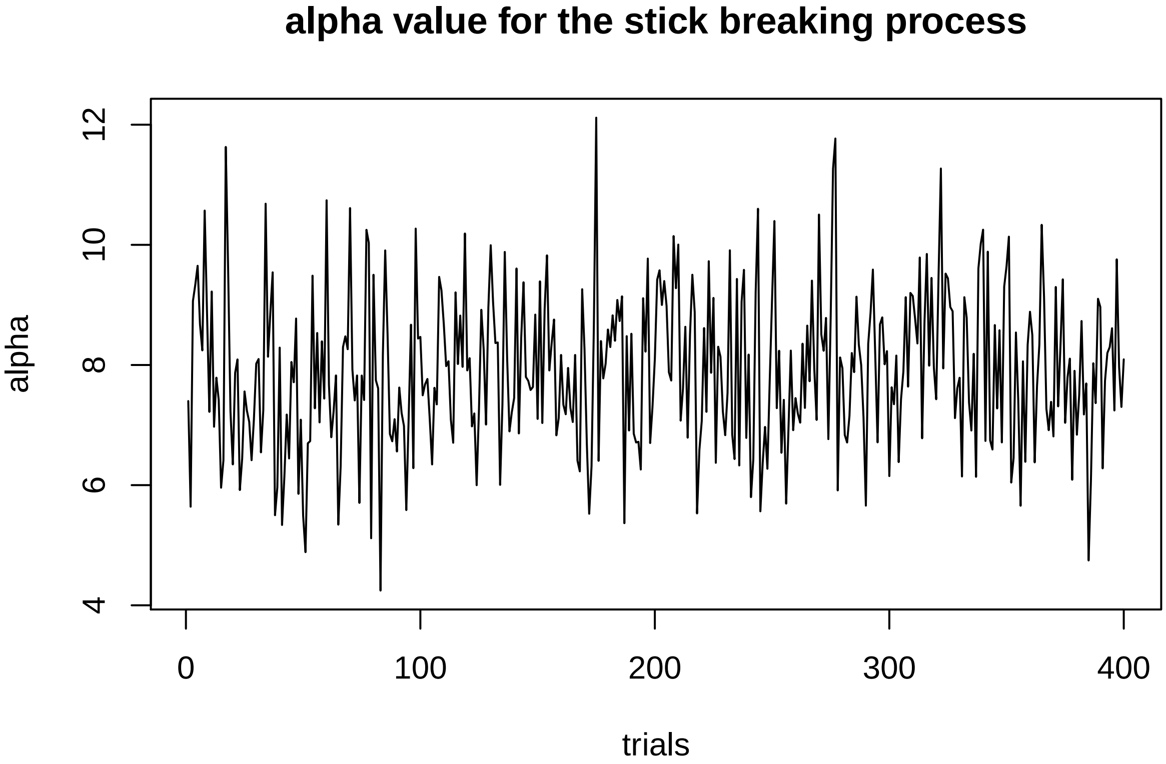


Blocked Gibbs Sampling Assessment: AGEP



Blocked Gibbs Sampling Assessment: PINCP


Number of clusters used over time
number of active clusters
0
10
20
30
40
0
100
200
300
400
trials


alpha value for the stick breaking process
alpha
4
6
8
10
12
0
100
200
300
400
trials