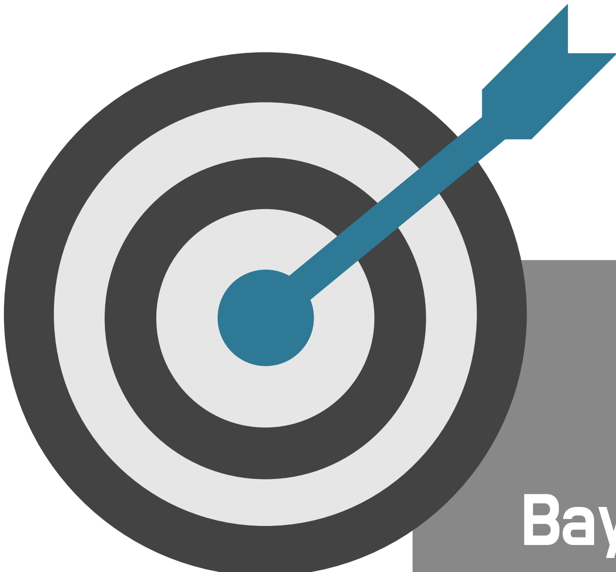
Crimes Imputation

using bayesian regression

2조

정여진
서민희
손동규
안재형
유아현

Project Goal



Bayesian inference를 활용하여
결측된 Y변수(범죄 발생 수) imputation

CONTENTS

1 EDA Summary

2 Model Selection

- Bayesian Regression
- Other Methods;
  - LASSO
  - SPCA

3 Conclusion

- Model Comparison
- Suggestions for Improvement
  - Mixture Distribution;
  - K-means Clustering

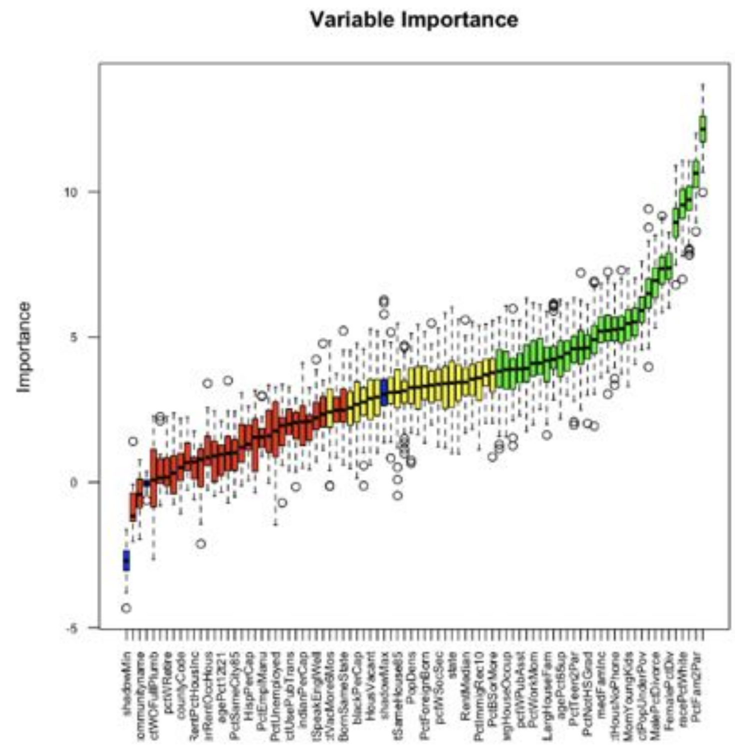
EDA

EDA Summary

Variable Selection – Boruta (Random Forest를 기반으로)

```r
# Do a tentative rough fix
roughFixMod <- TentativeRoughFix(boruta_output)
boruta_signif <- getSelectedAttributes(roughFixMod)
print(boruta_signif)
```

```
 [1] "state"               "population"          "householdsize"
 [4] "racepctblack"        "racePctWhite"        "racePctAsian"
 [7] "racePctHisp"         "agePct65up"          "medIncome"
[10] "pctWInvInc"          "pctWSocSec"          "pctWPubAsst"
[13] "medFamInc"           "perCapInc"           "blackPerCap"
[16] "PctPopUnderPov"      "PctLess9thGrade"     "PctNotHSGrad"
[19] "PctOccupManu"        "MalePctDivorce"      "FemalePctDiv"
[22] "PctFam2Par"          "PctKids2Par"         "PctYoungKids2Par"
[25] "PctTeen2Par"         "PctWorkMomYoungKids" "PctWorkMom"
[28] "PctKidsBornNeverMar" "NumImmig"            "PctRecImmig10"
[31] "PctLargHouseFam"     "PctLargHouseOccup"   "PersPerOccupHous"
[34] "PersPerOwnOccHous"   "PctPersOwnOccup"     "PctPersDenseHous"
[37] "PctHousLess3BR"      "HousVacant"          "PctHousNoPhone"
[40] "OwnOccMedVal"        "MedRent"             "PctForeignBorn"
[43] "PctSameHouse85"      "PopDens"
```



Hyperparameter를 default 값으로 돌린 결과 21개의 변수가 선택됨

EDA Summary

Variable Selection – Forward Stepwise Selection

```
# Show
print(shortlistedVars)
```

```
[1] "PctKids2Par"          "PersPerOwnOccHous"    "PctWorkMomYoungKids"
```

- 굉장히 적은 개수의 변수가 뽑히기 때문에 많은 정보를 상실

- 변수를 10-20개씩 group으로 나누어 stepwise를 다시 적용
- 선택된 세 가지 변수 모두 앞선 Boruta의 결과에 포함됨

EDA Summary

Variable Selection – Forward Stepwise Selection

Show

사전 EDA에서 유의미한 변수 선택을 하지 않고
Bayesian regression에서 선택된 변수 사용

EDA Summary

Divide into train and test set

Train Set

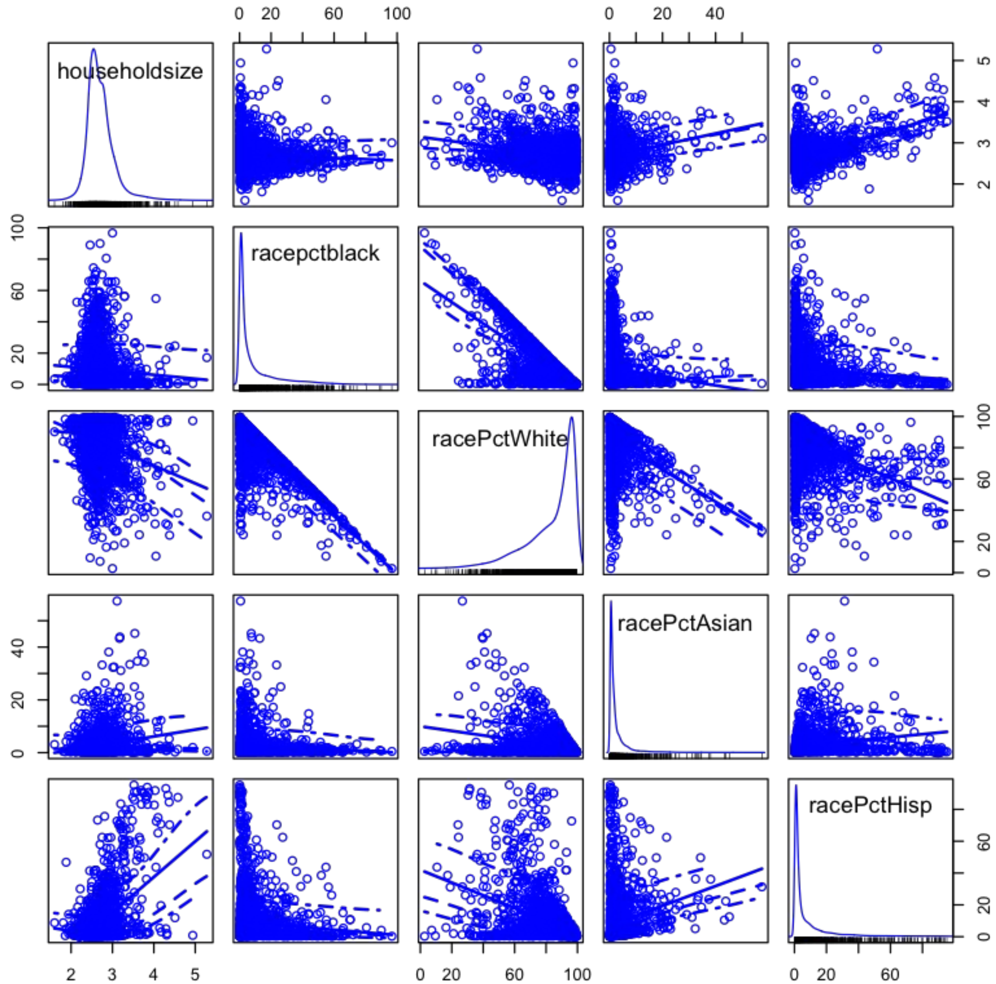
결측치가 존재하지 않는 row

Test Set

Y 변수 중 적어도 하나의 NA를 포함하고 있는 row (318개)

EDA Summary

Skewness and Scaling

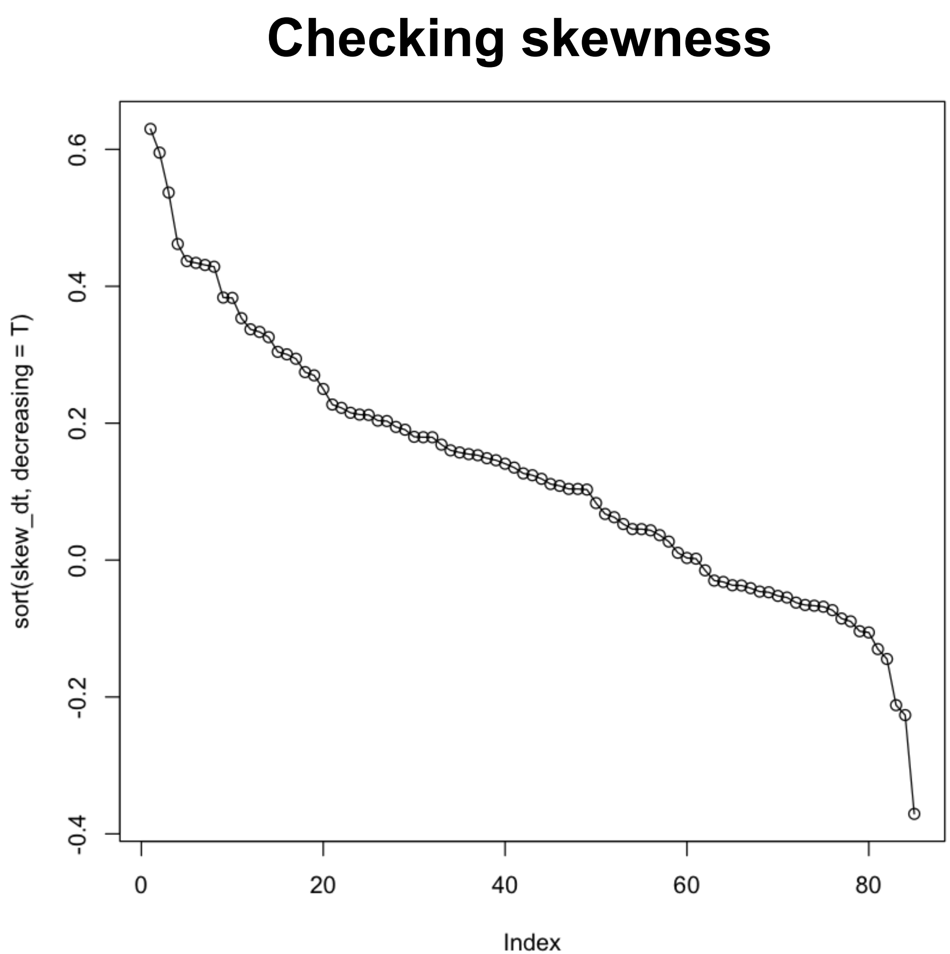


101개의 variables 중 model training에 필요 없는 variable들은 제외

Continuous variables에 한해 scaling, skewness 조정 후 MedNumBR과 붙임

EDA Summary

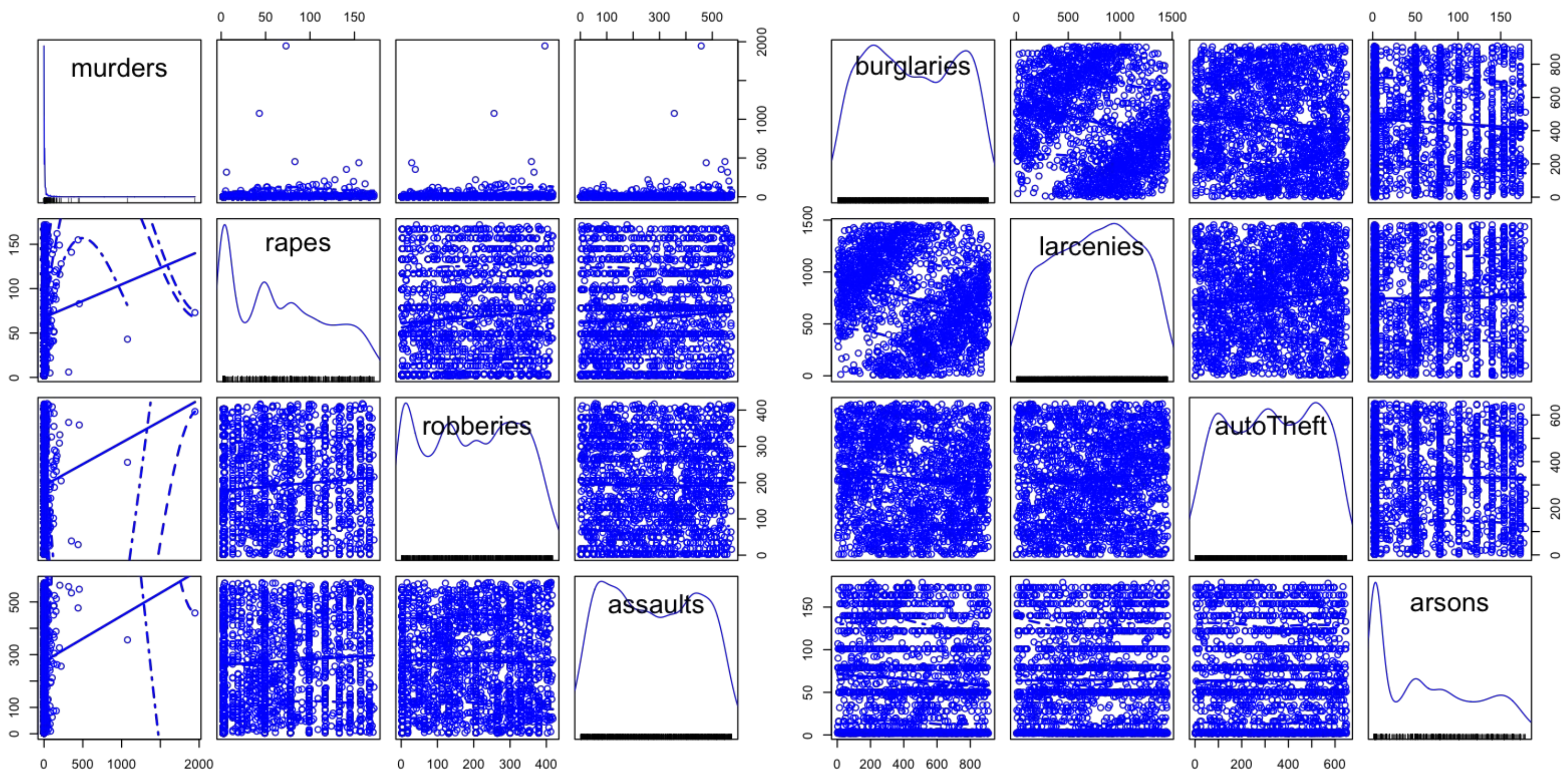
Skewness and Scaling



EDA Summary

Skewness and Scaling

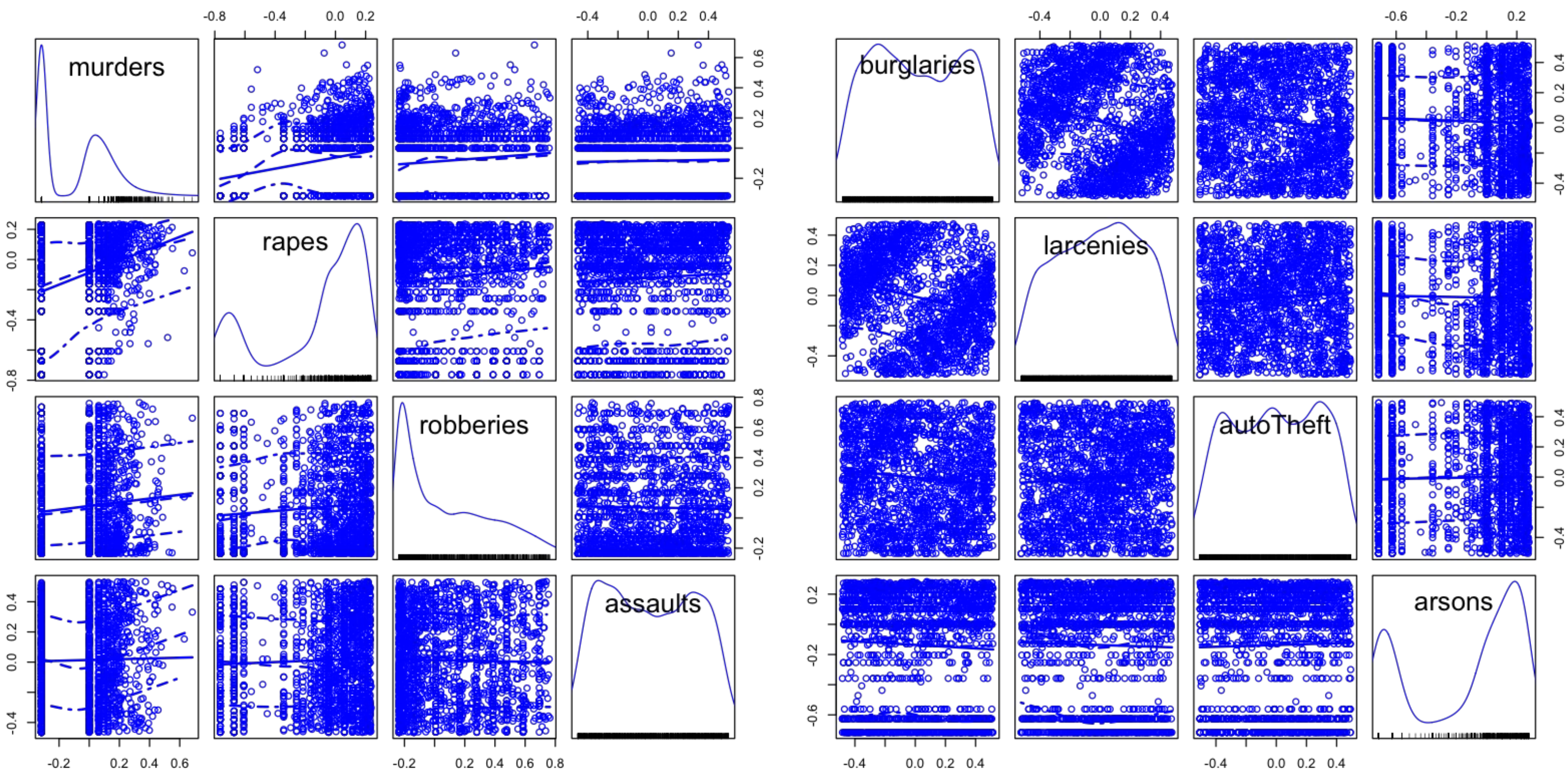
Scatter plot of training set



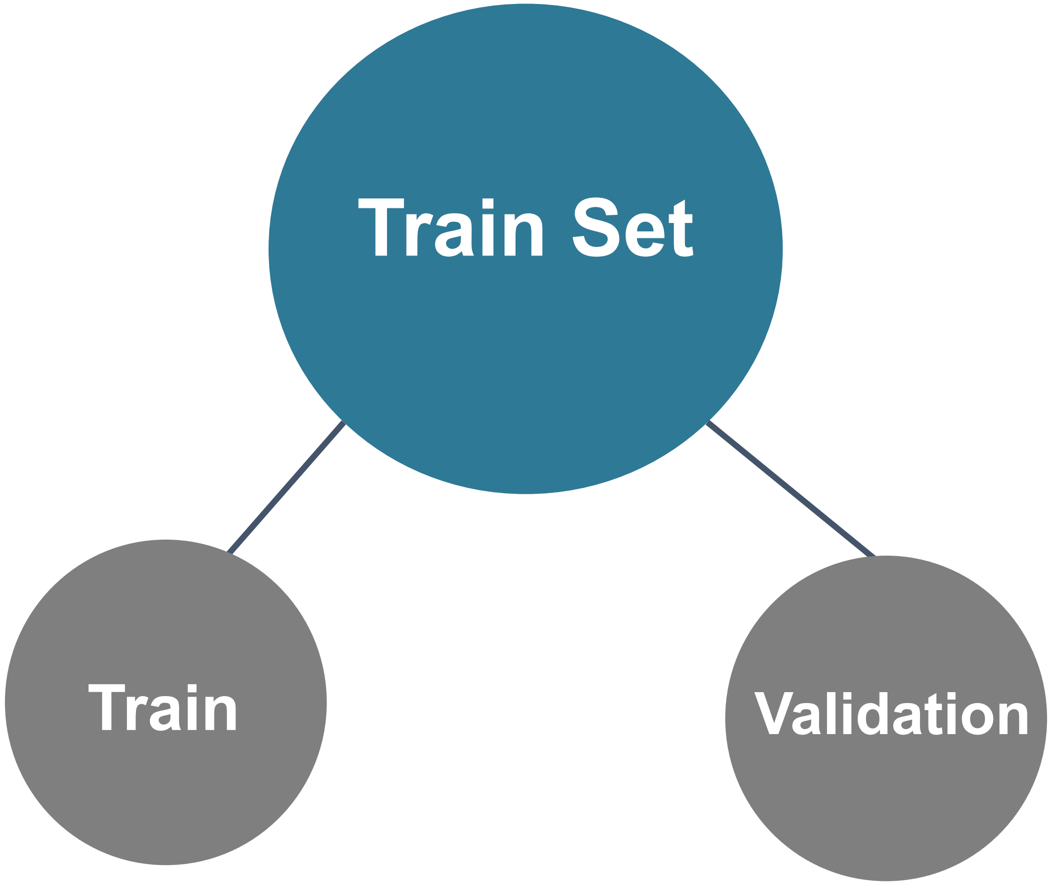
EDA Summary

Skewness and Scaling

Scatter plot of scaled and skewed traing set



EDA Summary



Start variable selection

EDA Summary

Murders

Rapes

Robberies

Assaults

Burglaries

Larcenies

AutoTheft

Arsons

Bayesian regression

Bayesian regression

i) Bayesian regression code -'bayes_reg'

```r
#Bayes Selection function
#input is target index(1~8)

bayes_reg=function(target,sample_n){
    t=which(colnames(scaled.skew.t)==target)
    Y=as.matrix(scaled.skew.t[,t]); colnames(Y)= 'Y' ;N=length(Y)
    crim.lse = lm(Y~. -1, data=as.data.frame(cbind(Y, X))) # Y~. -1 as X already includes an intercept col
    MSE = sum(crim.lse$residuals^2) / (N-p)


    #prior parameters
    g=N
    v0 = 1; s20 = MSE


    #log likelihood of data
    lpy.X = function(Y, X, g=length(Y), v0=1, s20 = try(summary(lm(Y~-1+X))$sigma^2, silent = T)){
        n = dim(X)[1]; p =dim(X)[2]
        if(p==0) {Hg = 0; s20 = mean(Y^2)} # if a model with no regressor is selected
        if(p>0) {Hg = g/(g+1) * X %*% solve(t(X)%*%X) %*% t(X)}
        SSRg = t(Y) %*% ( diag(1, nrow=n) - Hg ) %*% Y
        return(-.5*(n*log(pi)+p*log(1+g)-v0*log(v0*s20)+(v0+n)*log(v0*s20+SSRg))
               +lgamma((v0+n)/2)-lgamma(v0/2))
    }


    #Gibbs sampling - z
    S=sample_n
    z = rep(1, dim(X)[2])            # initial value for the model string Z
    lpy.c = lpy.X(Y,X[,z==1, drop=F])  # calculate current logP(Y|X,z)
    Z = matrix(NA, S, dim(X)[2])       # result slot
    M_lpy = matrix(rep(NA, S),ncol=1)  # result slot (optional)

  for(s in 1:S){ progress(s, S-1)
              for(j in sample(1:dim(X)[2])){
                  zp = z; zp[j] = 1 - zp[j] # change 1 to 0, 0 to 1
                  lpy.p = lpy.X(Y,X[,zp==1, drop=F]) # calculate logP(Y|Xz) for proposed z
                  r = (lpy.c - lpy.p)*(-1)^(zp[j]==0)
                  z[j] = rbinom(1,1,1/(1+exp(r))) # change 1 to 0, 0 to 1 with prob. 1/(1+exp(r))
                  if(z[j] == zp[j]){lpy.c = lpy.p}
              }
              Z[s,] = z; M_lpy[s] = lpy.c
             }

    #MCMC - beta, sigma
    SIGMA = matrix(nrow=S, ncol=1); BETA = matrix(nrow=S, ncol=dim(X)[2])
    for(s in 1:S){
        Xz = X[,Z[s,]==1, drop=F]

        # sigma given Y, X
        Hg = (g/(g+1)) * Xz %*% solve(t(Xz)%*%Xz) %*% t(Xz)
        SSRg = t(Y) %*% (diag(1, dim(Xz)[1]) - Hg) %*% Y
        SIGMA[s,] = 1/rgamma(1, (v0+N)/2, (v0*s20 + SSRg)/2)

        # beta given Y, X, sigma
        Vb = (g/(g+1)) * solve(t(Xz) %*% Xz) *SIGMA[s, ]
        Eb = (g/(g+1)) * solve(t(Xz) %*% Xz) %*% t(Xz) %*% Y
        BETA[s,Z[s,]==1] = mvrnorm(1, Eb, Vb)
    }

      #output - beta / cutoff=0.5
      Z_post_mean = apply(Z, 2, mean, na.rm=T)
      Beta_Bay_est = apply(BETA, 2, mean, na.rm=T)
      final_z=Z_post_mean[Z_post_mean>0.3]
      final_z_index=which(Z_post_mean>0.3)
      final_beta=Beta_Bay_est[final_z_index]

      output_dt=as.data.frame(cbind(final_z,final_beta))
      rownames(output_dt)=colnames(X[,final_z_index])
      colnames(output_dt)=c('final_z','final_Beta')

      newlist=list(output_dt=output_dt, Beta_Bay_est=Beta_Bay_est)
      return(newlist)
}
```

Bayesian regression

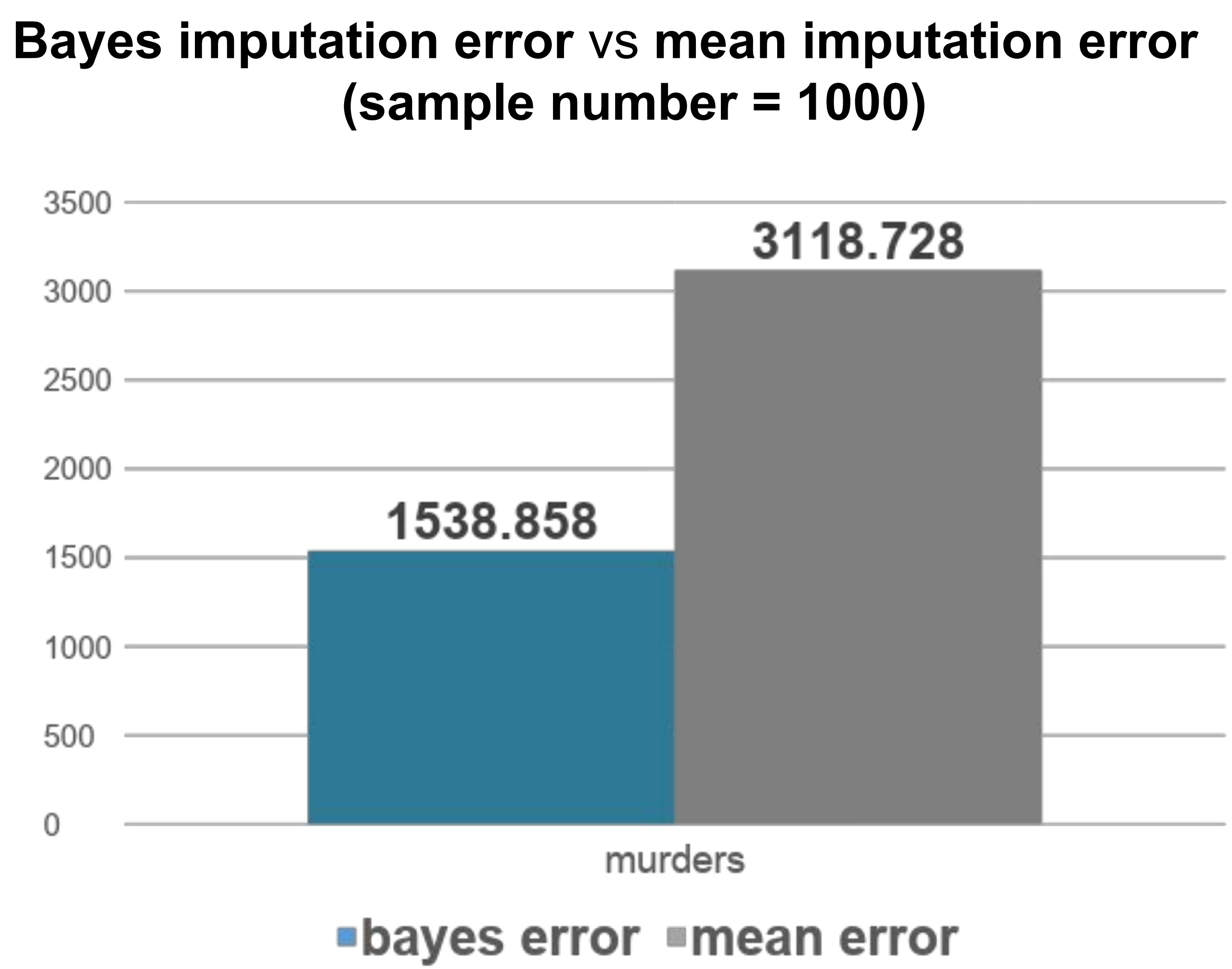
ii) Bayesian error code -'bayes_err'

```r
##Real values are stored in 'real.v'
##Scaled and skew-adjusted values are stored in 'scaled.skew.v'
##1. Define coefficient vector
##2. Define design matrix
##3. Predict: X %*% coefficient
##4. Transform predictions: unscale ->log,square (skew.train.set)
##5. Compare with real.v
set.seed(111)

bayes_err=function(target, sample_n, skew_trans){

    output_list=bayes_reg(target,sample_n)

    ##0. Conduct Bayes Selection
    t=which(colnames(scaled.skew.t)==target)
    output_crimes=output_list[[1]]
    output_crimes=as.data.frame(output_crimes)
    index_crimes=which(colnames(scaled.skew.v) %in% rownames(output_crimes))

    ##1. Define coefficient vector
    Beta_Bayes_est=output_list[[2]]
    {if(sum(rownames(output_crimes) %in%'intercept')==1) coef_crimes<-output_crimes[,2]
     else if(Beta_Bayes_est[1]=='NaN') coef_crimes<-output_crimes[,2]
         else coef_crimes<-c(Beta_Bayes_est[1],output_crimes[,2])
    }

    ##2. Define design matrix
    X_crimes=as.matrix(cbind(1,scaled.skew.v[,index_crimes]))
    if(ncol(X_crimes)==length(coef_crimes)){
        X_crimes<-X_crimes
    }else X_crimes<-X_crimes[,-1]

    ##3. Predict: X %*% coefficient
    pred_crimes=X_crimes %*% coef_crimes

    ##4. Transform predictions: unscale ->log,square (skew.train.set)
    pred_crimes_real=pred_crimes*(max(skew.train.set[,t])-min(skew.train.set[,t]))+median(skew.train.set[,t])
    {if(skew_trans=='log') pred_crimes_real<-exp(pred_crimes_real)
     else if(skew_trans=='sq') pred_crimes_real<-sqrt(pred_crimes_real)
         else pred_crimes_real<-pred_crimes_real
    }

    #pred_crimes_real= ifelse(skew_trans=='log',exp(pred_crimes_real),
    #       ifelse(skew_trans=='sq',sqrt(pred_crimes_real),pred_crimes_real))


    ##5. Compare with real.v
    err=sum((pred_crimes_real-real.v[,t])^2)/length(real.v)
    newlist2=list(output_crimes,err)

    return(newlist2)
    }
```

bayes_err(범죄명, sample 수, transform 형태)

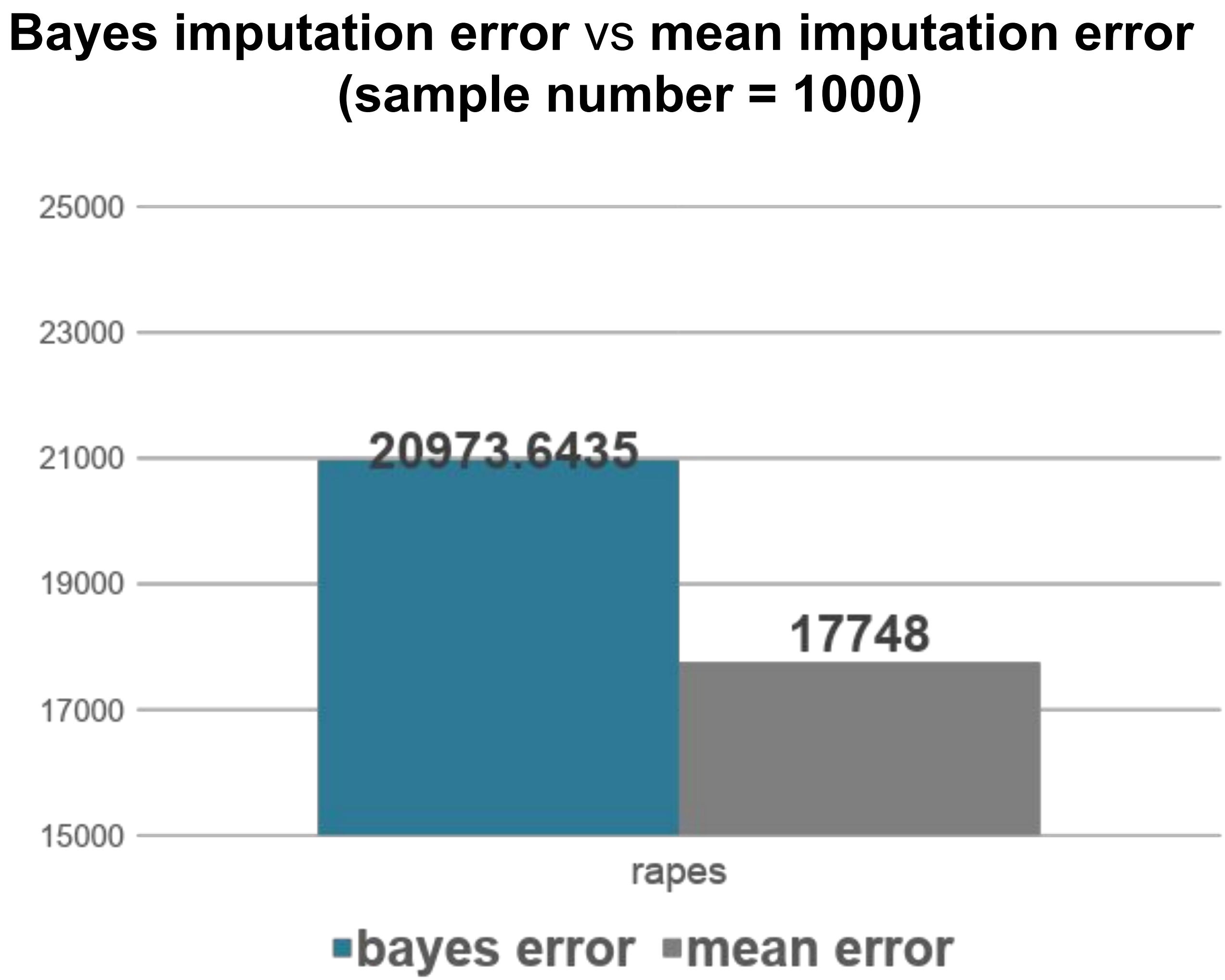
Bayesian regression

1. murders



Bayesian regression

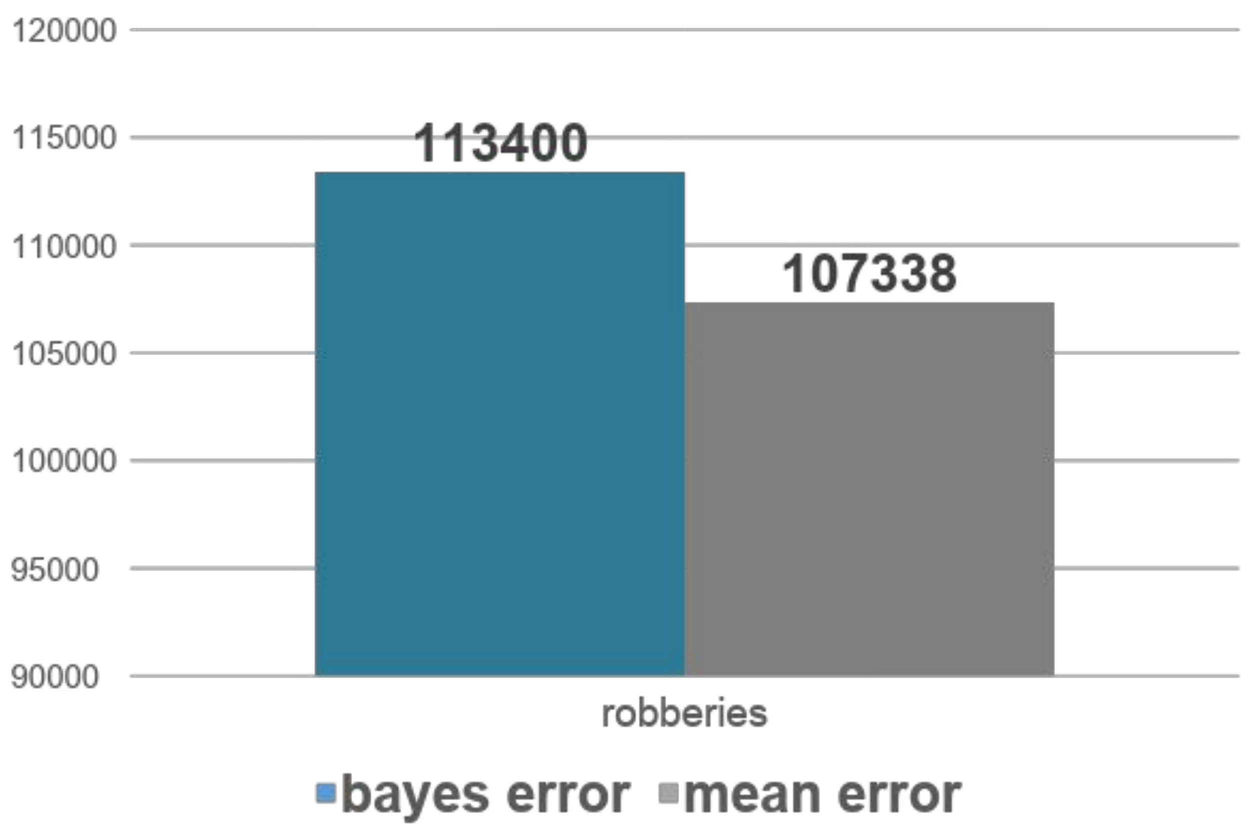
2. rapes



Bayesian regression

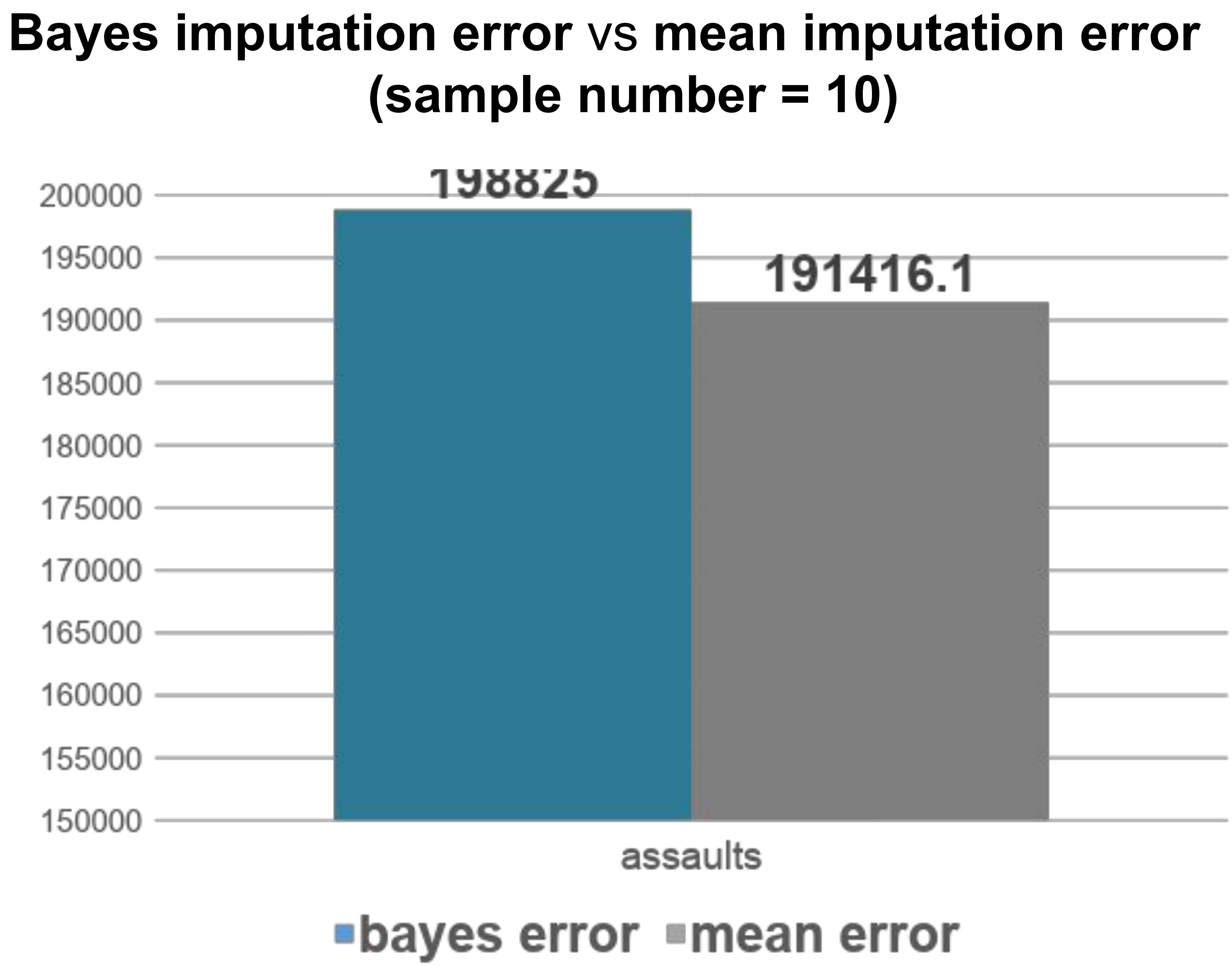
3. robberies

Bayes imputation error vs **mean imputation error (sample number = 1000)**



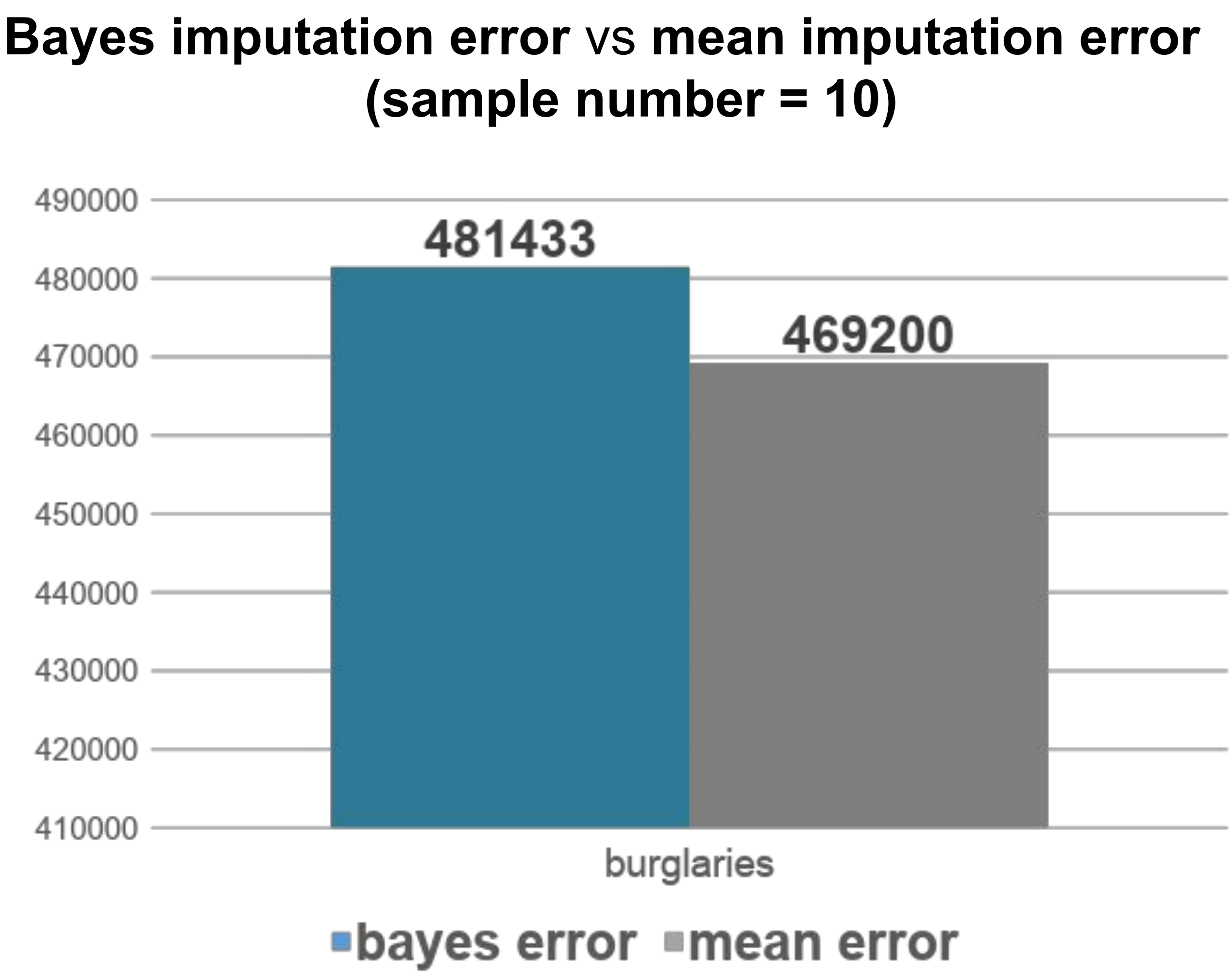
Bayesian regression

4. assaults



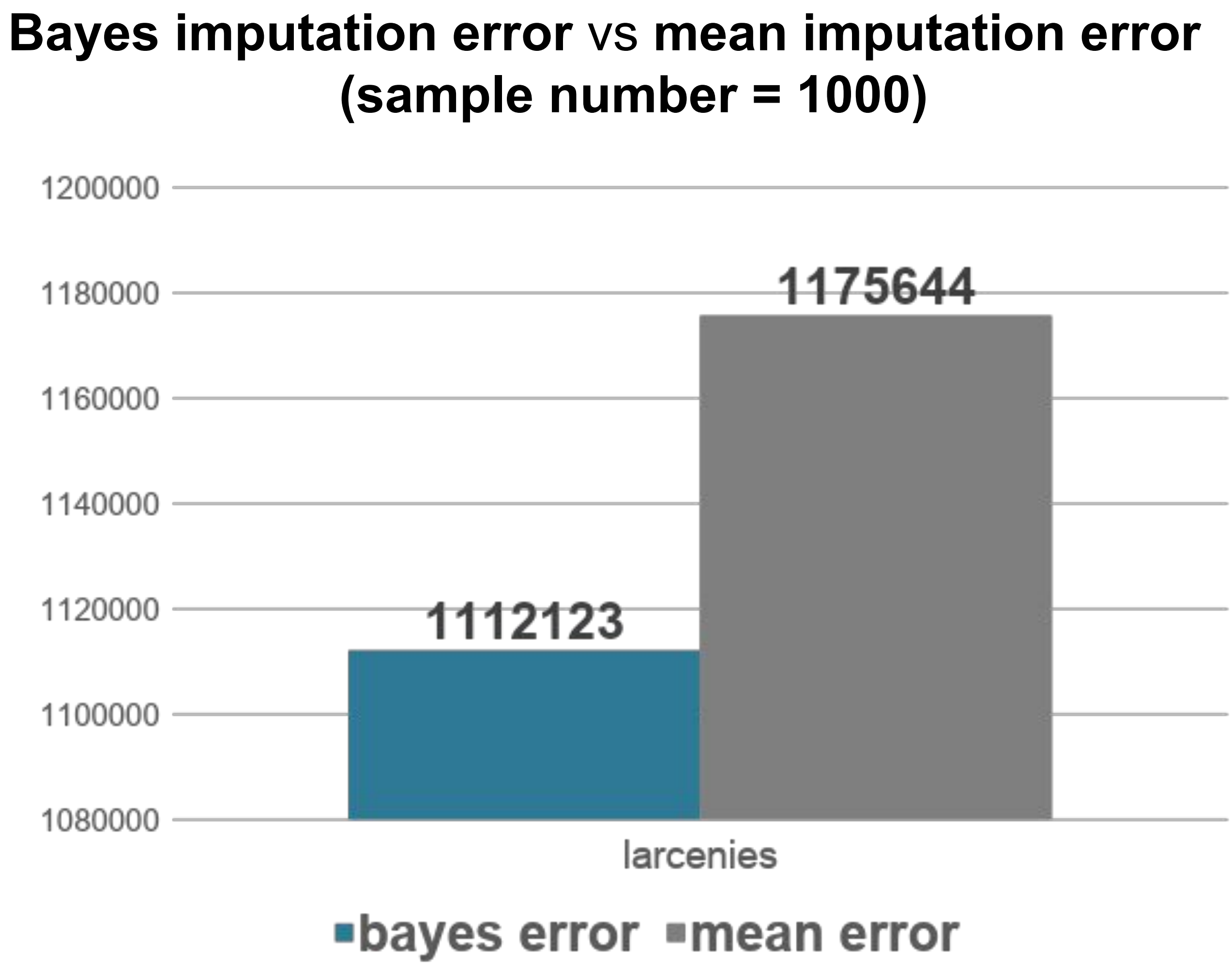
Bayesian regression

5. burglaries



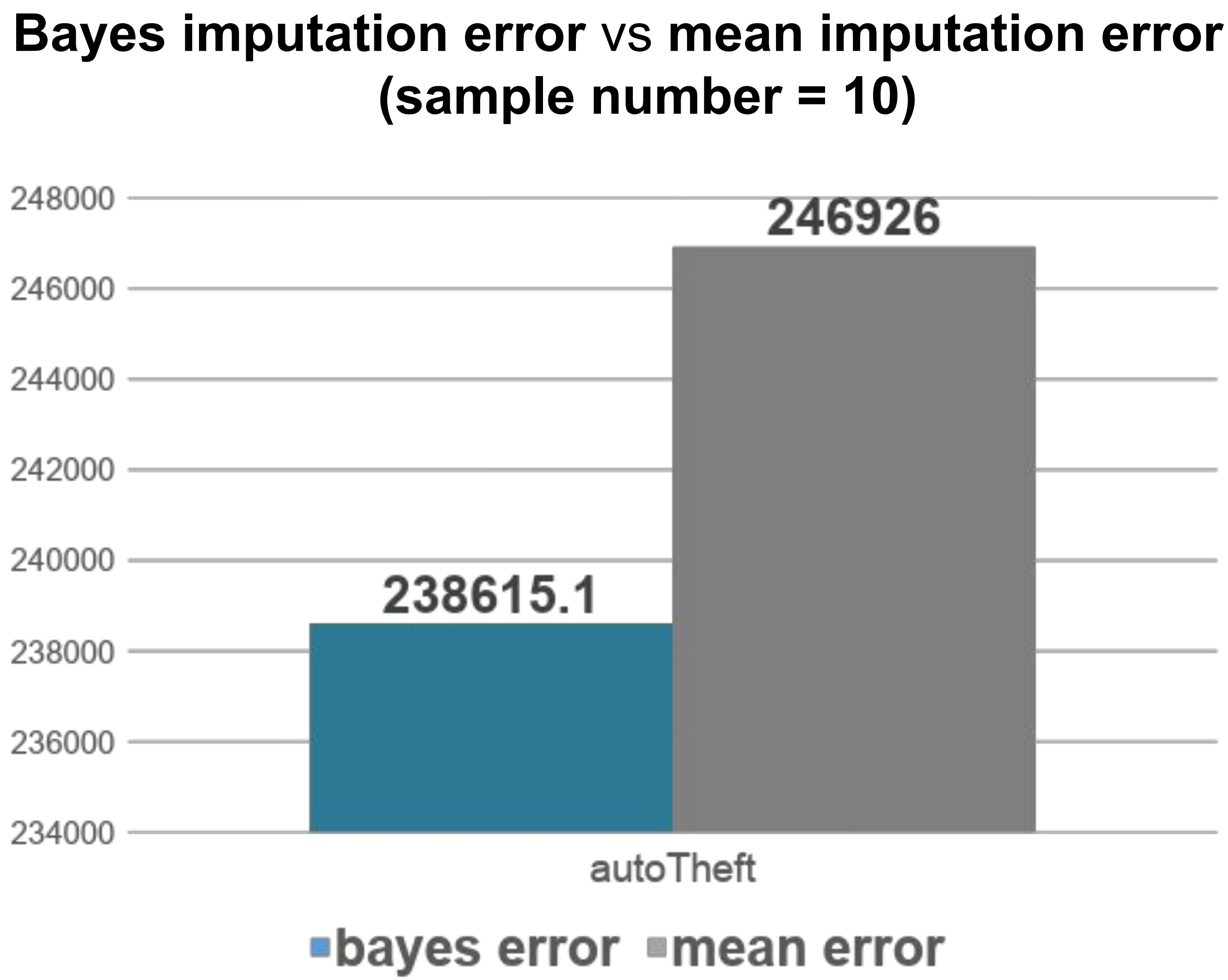
Bayesian regression

6. larcenies



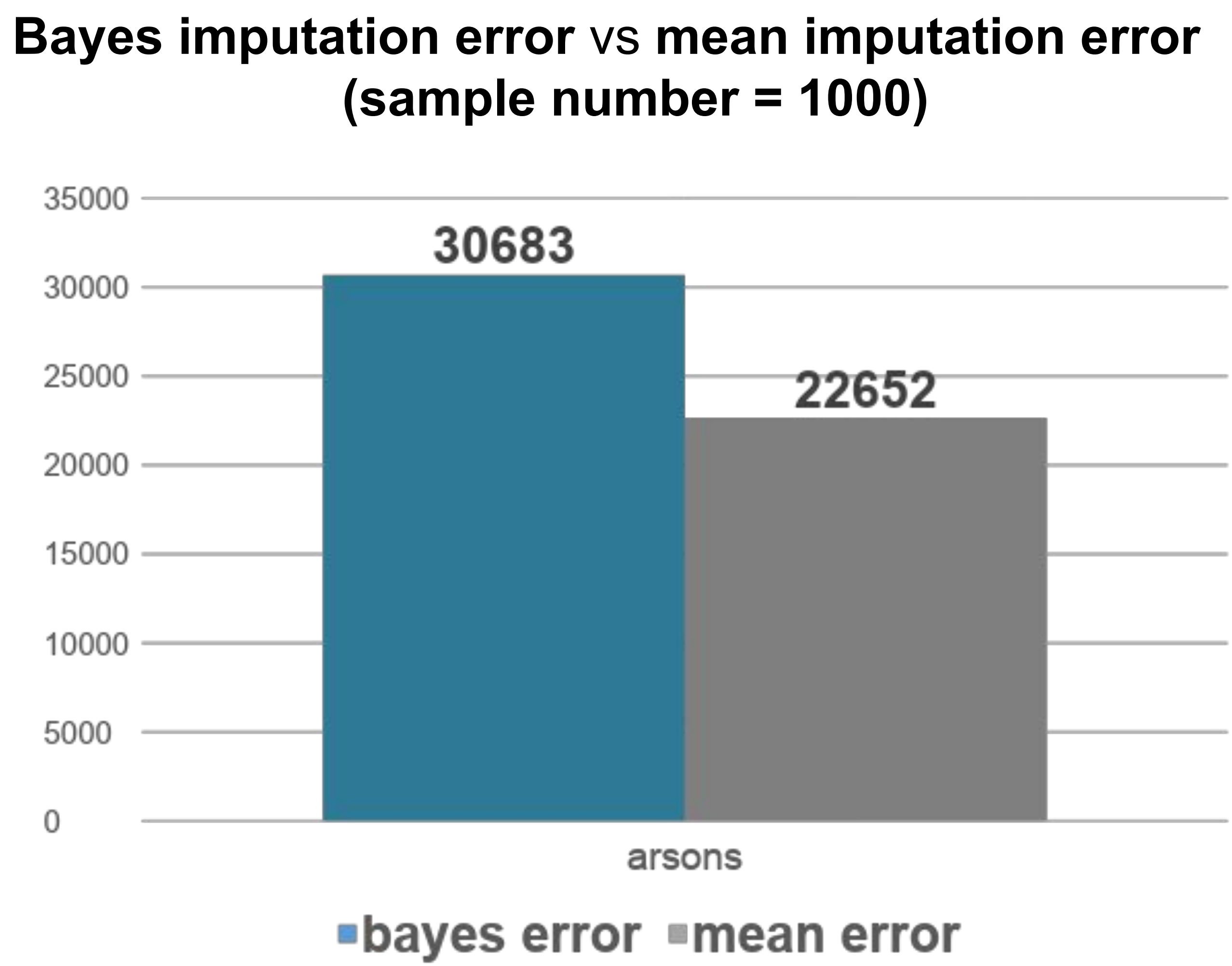
Bayesian regression

7. autoTheft



Bayesian regression

8. arsons



Other Methods – LASSO

LASSO

Lasso(Least Absolute Shrinkage and Selection Operator) 란?

기존의 Linear Regression에서 적절한 가중치와 편향을 찾아내는 것이 관건이었다면, LASSO(Least Absolute Shrinkage and Selection Operator)는 거기에 덧붙여서 추가 제약조건(L1 Norm)을 준다. 그 제약조건은 MSE가 최소가 되게 하는 가중치와 편향을 찾는 데 동시에 가중치들의 절대값들의 합, 즉 가중치의 절대값들이 최소가 되게 해야한다는 것이다. 다시 말해서 가중치의 모든 원소가 0이 되거나 0에 가깝게 되게 하는 것이다.

LASSO

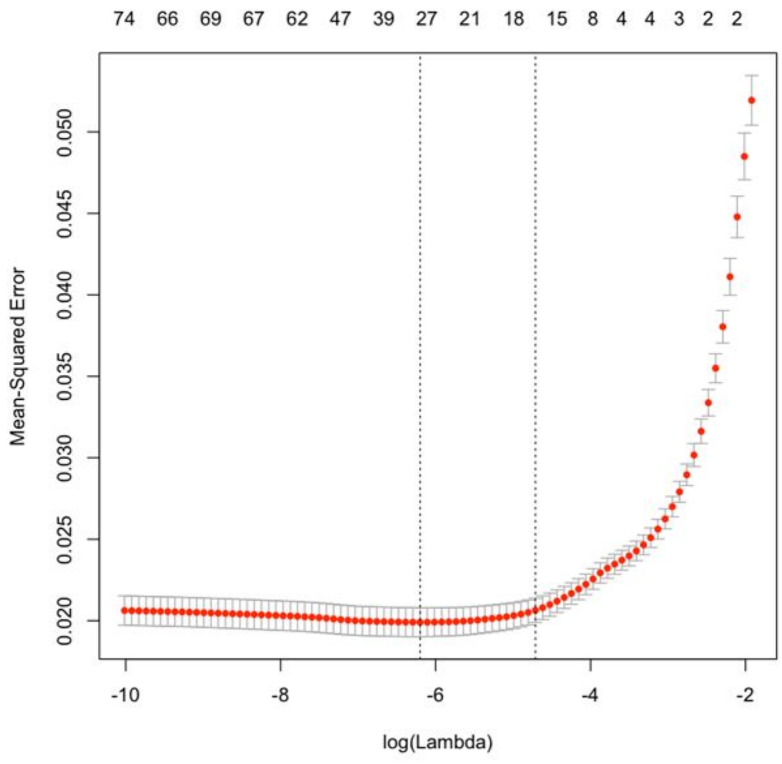
Lasso(Least Absolute Shrinkage and Selection Operator) 란?

$$
\begin{aligned}
&MSE \;+\; penalty \\
&= MSE \;+\; \alpha \cdot L_1\text{-}norm \\
&= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \;+\; \boxed{\alpha \sum_{j=1}^{m} |w_j|}
\end{aligned}
$$

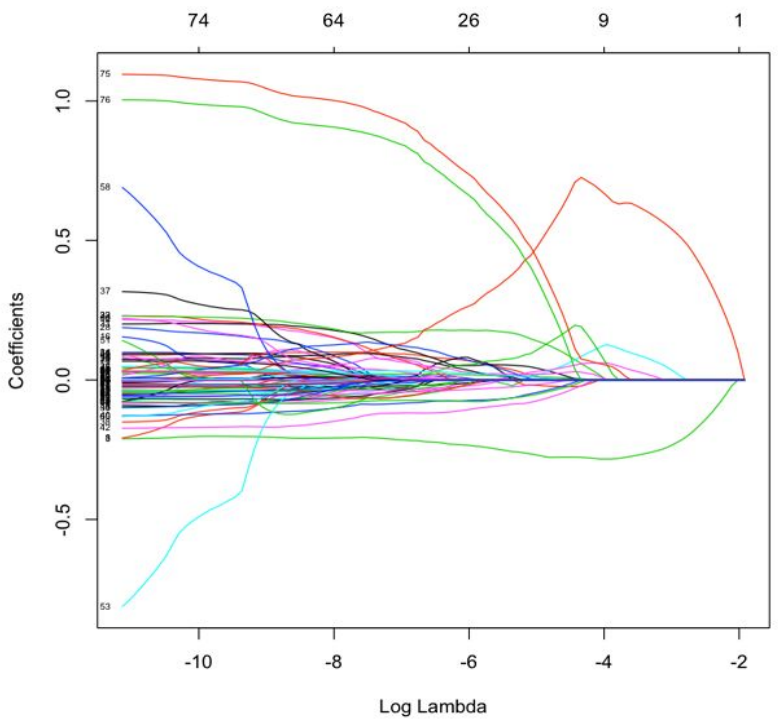
L1 Penalty

LASSO

< Cross Validation을 통한 패널티 가중치 추정>



< 몇몇 계수들이 0으로 shrink>



LASSO

Robberies - sq

In [111]:

```
X <- as.matrix(scaled.skew.t[, -c(78:85)])
cv3 <- cv.glmnet(X, scaled.skew.t[[80]],alpha=1)

cv.lasso=cv3
lasso.coef = predict(cv.lasso, type = "coefficients", s=cv.lasso$lambda.min) # output=sparse matrix
lasso.coef[,1][lasso.coef[,1]!=0]
length(lasso.coef[,1][lasso.coef[,1]!=0])

lasso.test=as.matrix(scaled.skew.v[,-c(78:85)])
lasso.pred3 = predict(cv.lasso, s=cv.lasso$lambda.min, newx = lasso.test)

lasso_pred_real3=lasso.pred3*(max(skew.train.set[,80])-min(skew.train.set[,80]))+median(skew.train.set[,80])
lasso_pred_real3[lasso_pred_real3<0]<-0
lasso_pred_real3=sqrt(lasso_pred_real3)
err_l3=sum((lasso_pred_real3-real.v[,80])^2)/length(real.v)
err_l3
```

(Intercept)	0.0533646760907696
householdsize	-0.0725768189916002
racepctblack	0.0673734278588427
racePctWhite	-0.047254749872441
agePct12t21	-0.0113046911085456
indianPerCap	0.000741025051785238
OtherPerCap	-0.0279703274022045
PctWorkMomYoungKi...	-0.0181814396815018
PctWorkMom	-0.0269957828894548
PctNotSpeakEnglWell	0.106059912575755
HousVacant	0.019157311184707
MedYrHousBuilt	-0.0356171837452312
MedRentPctHousInc	-0.0859597487008759
PopDens	0.095090494946478

대략 20~30개의 변수가 뽑힘

Other Methods – Sparse PCA

Sparse PCA

SPCA(Sparse Principal Component Analysis)란?

기존 PCA의 principal components는 모든 input variable의 linear combination이기 때문에 해석의 어려움이 발생

반면,
Sparse PCA의 principal components는 유의한 input variable의 linear combination

SPCA는 기존 PCA에 LASSO penalty를 접목,
data의 dimension을 줄임

⇨ input variable이 많은 high-dimensional data분석에 용이

Sparse PCA

1.

medIncome	-0.46538576497294
PctPopUnderPov	0.659971542326747
PctHousNoPhone	0.31298619142701
OwnOccMedVal	-0.21616843495342
MedRent	-0.450737733926516

Loadings of PC1

racePctWhite	0.472732222887109
alePctDivorce	-0.0304452226885781
PctKids2Par	0.872375347984423
ornNeverMar	-0.119479774139656
rsDenseHous	-0.016827645713195

2.

racePctHisp	-0.332920000282132
NumImmig	-0.622739124884626
PctRecImmig10	-0.701426117206124
PctBornSameState	0.072587694995952
PctUsePubTrans	-0.0639740915245425

ctNotHSGrad	-0.00894006241440504
PctBSorMore	0.782459647714106
PctEmplManu	-0.0419645047027797
EmplProfServ	0.411262787750252
ctOccupManu	-0.465595183329988

3.

MedYrHousBuilt	-0.807654893837169
PctBornSameState	0.388960893927252
PctSameHouse85	0.205911657532864
PctSameCity85	0.353840254551932
PctUsePubTrans	0.169706979841363

HousVacant	-0.146429124872914
ctHousOccup	0.0697599661948257
acantBoarded	-0.684899639952924
VacMore6Mos	-0.688682572092002
LandArea	-0.174128850551965

4.

MalePctDivorce	0.113952089419724
PctLargHouseOccup	-0.531599324332032
PersPerOccupHous	-0.825451483370874
PersPerRentOccHous	-0.141274462277511
PctBornSameState	-0.055573866120957

racePctHisp	0.01550348573795
indianPerCap	0.975917578086567
alePctDivorce	0.196249360851691
FemalePctDiv	0.0889681575797726
HousVacant	0.030255214654448

5.

racePctHisp	-0.023973850024768
OtherPerCap	-0.999248815201844
PctPopUnderPov	-0.0237752537878255
PctPersDenseHous	-0.0162385920708983
PctSameHouse85	0.00990481032533901

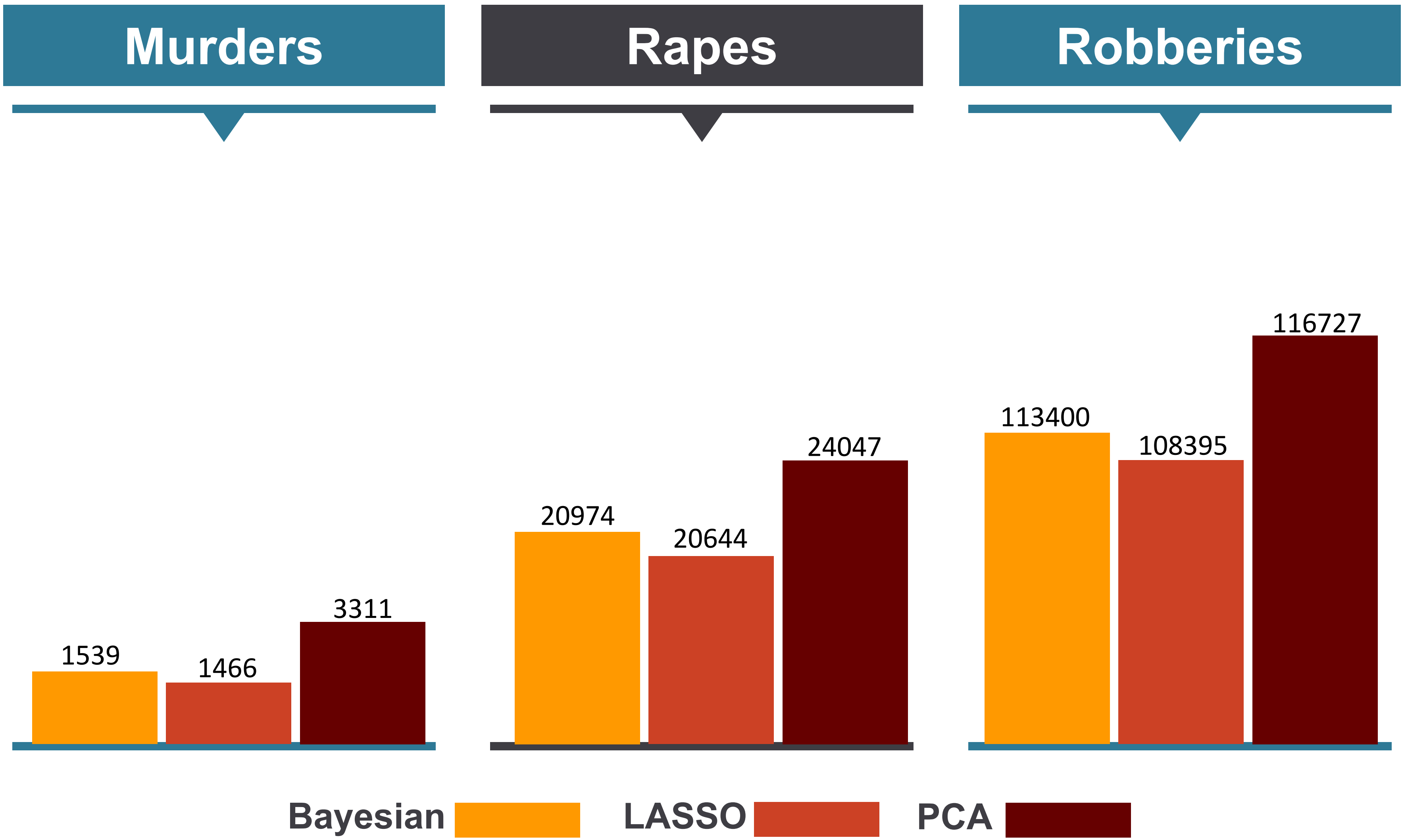
pctWWage	0.297927248752846
pctWSocSec	-0.733813927224357
PctEmploy	0.43767524210203
tImmigRec10	0.425641171216214
ctSameCity85	-0.00514300289171929

PC 개수는 10개, 한 PC당 5개 변수

Conclusion – Model Selection

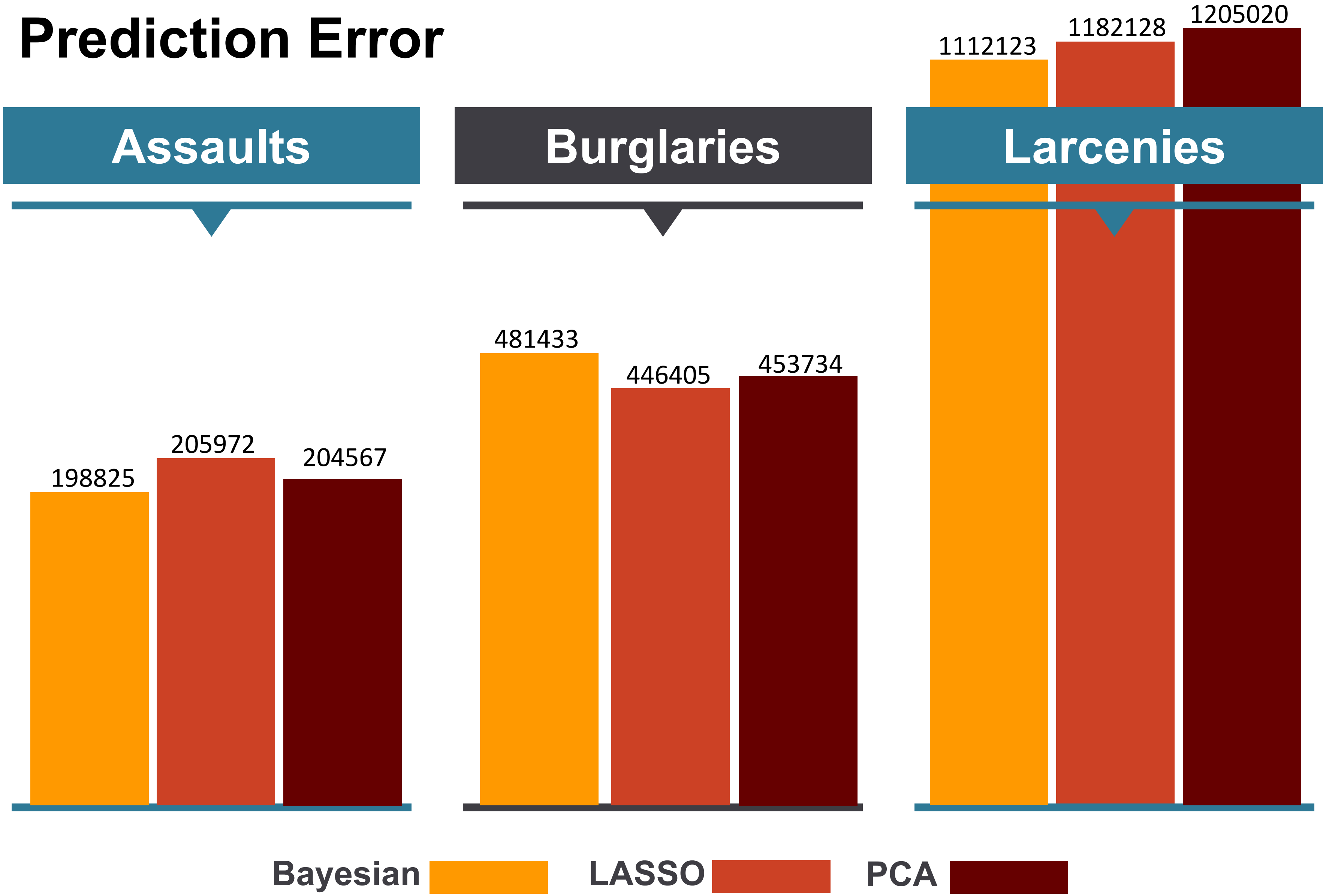
Model Selection

Prediction Error



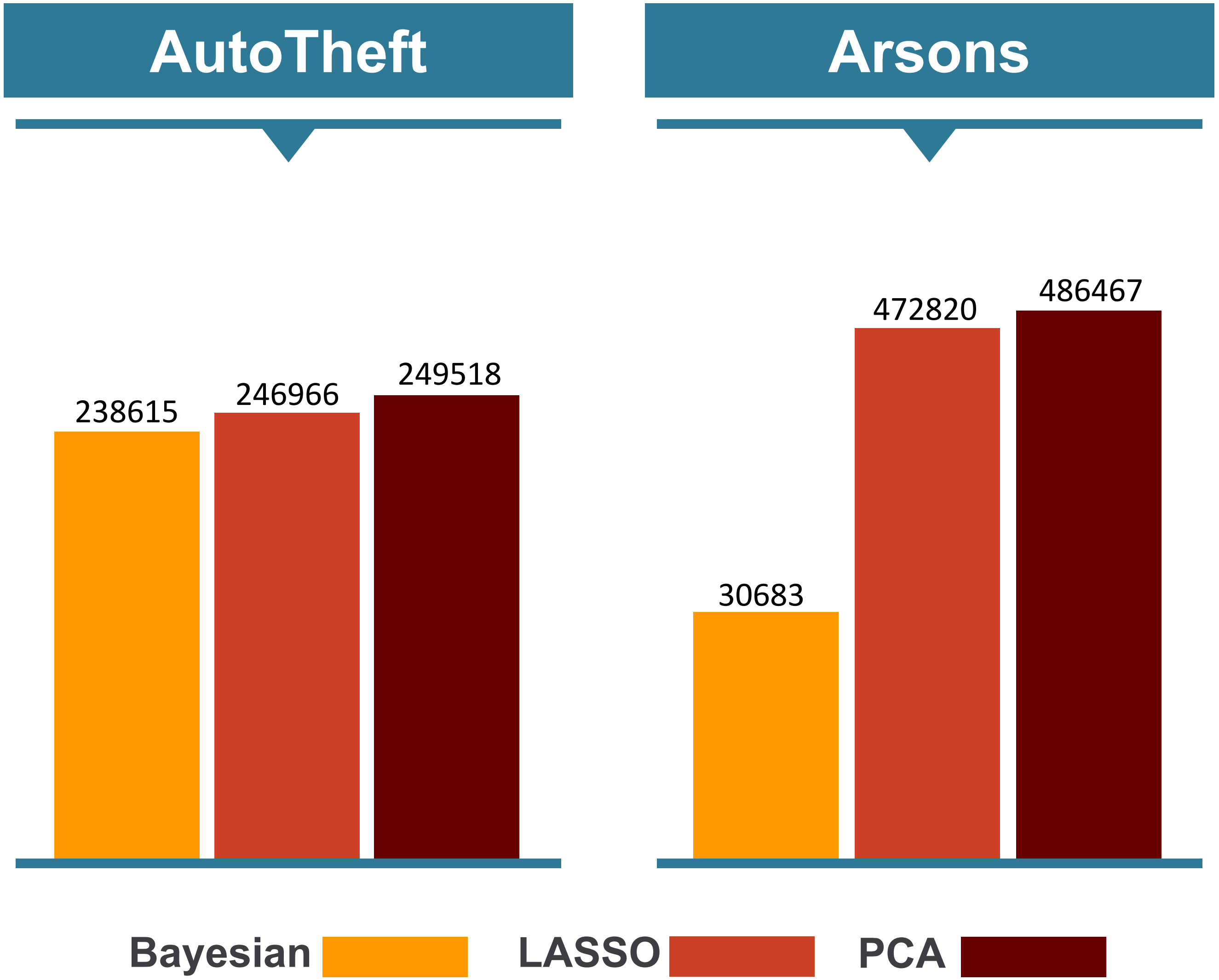
Model Selection

Prediction Error



Model Selection

Prediction Error



Model Selection

Our Models

Bayesian Reg	LASSO	PCA
4	4	0
Assaults Larcanies / AutoTheft/ Arsons	Murders / Rapes / Robberies Burglaries	

최종 모델은 Bayesian Regression으로 선택함

Conclusion – Suggestions

K-means Clustering
for Mixture Distribution

Murders

Assaults

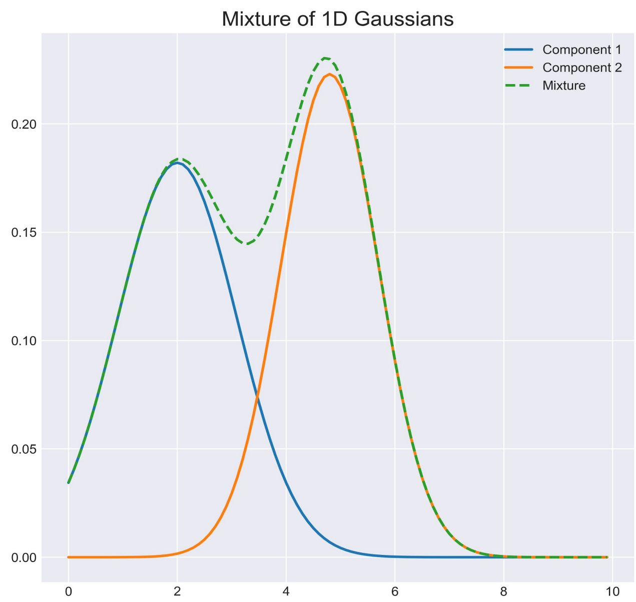
burglaries

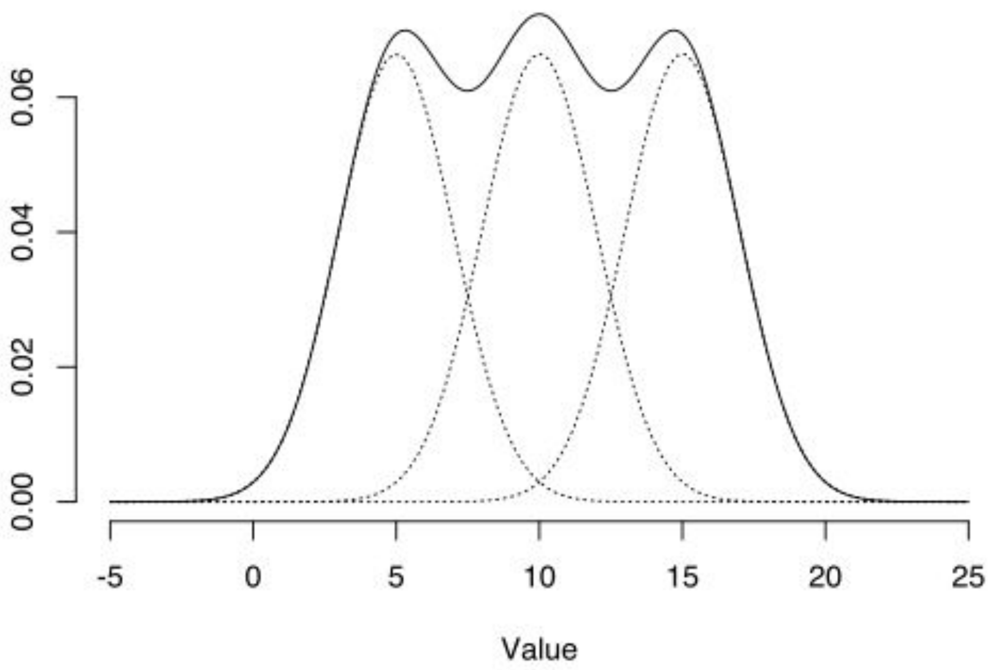
AutoTheft

K-means Clustering
for Mixture Distribution

Mixture Gaussian Distributions Detected

- 하나의 정규분포 형태가 아닌 문제 발생





K-means Clustering
for Mixture Distribution

Mixture Gaussian Distributions Detected

- 하나의 정규분포 형태가 아닌 문제 발생

9.2 Bayesian estimation for a regression model

We begin with a simple semiconjugate prior distribution for $\boldsymbol{\beta}$ and σ^2 to be used when there is information available about the parameters. In situations where prior information is unavailable or difficult to quantify, an alternative "default" class of prior distributions is given.

9.2.1 A semiconjugate prior distribution

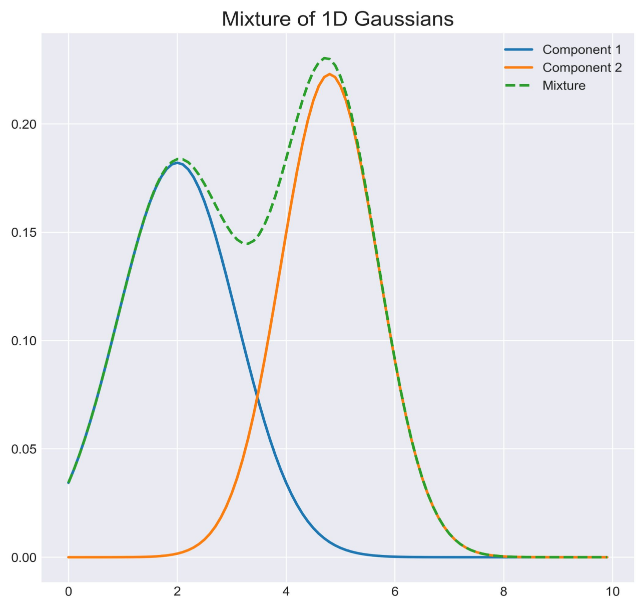
The sampling density of the data (Equation 9.3), as a function of $\boldsymbol{\beta}$, is

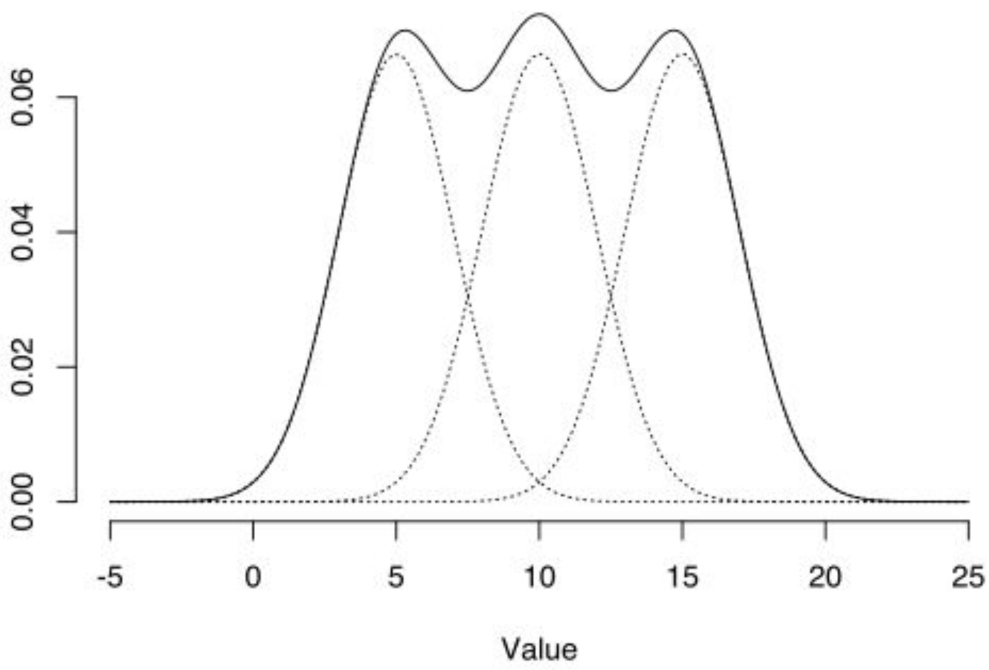
$$\boxed{\begin{aligned} p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\beta}, \sigma^2) &\propto \exp\{-\frac{1}{2\sigma^2}\mathrm{SSR}(\boldsymbol{\beta})\} \\ &= \exp\{-\frac{1}{2\sigma^2}[\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\boldsymbol{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}]\}. \end{aligned}}$$

K-means Clustering
for Mixture Distribution

Mixture Gaussian Distributions Detected

- 해결책: Y를 cluster로 나누어 cluster별 특징 확인

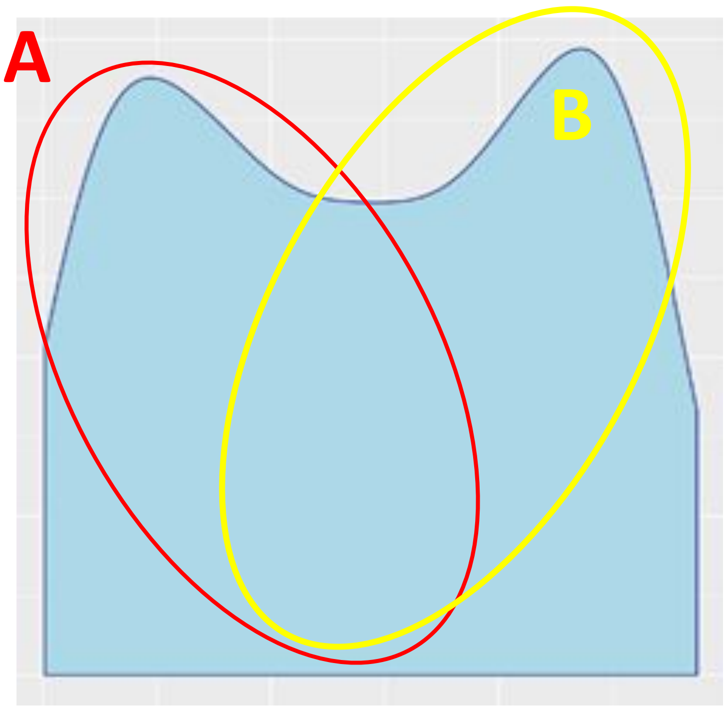




K-means Clustering
for Mixture Distribution

Mixture Gaussian Distributions Detected

- 해결책: Y를 cluster로 나누어 cluster별 특징 확인



'Assaults'

In [64]:
```
valid=s.train.set[index,]
traini=s.train.set[-index,]
```

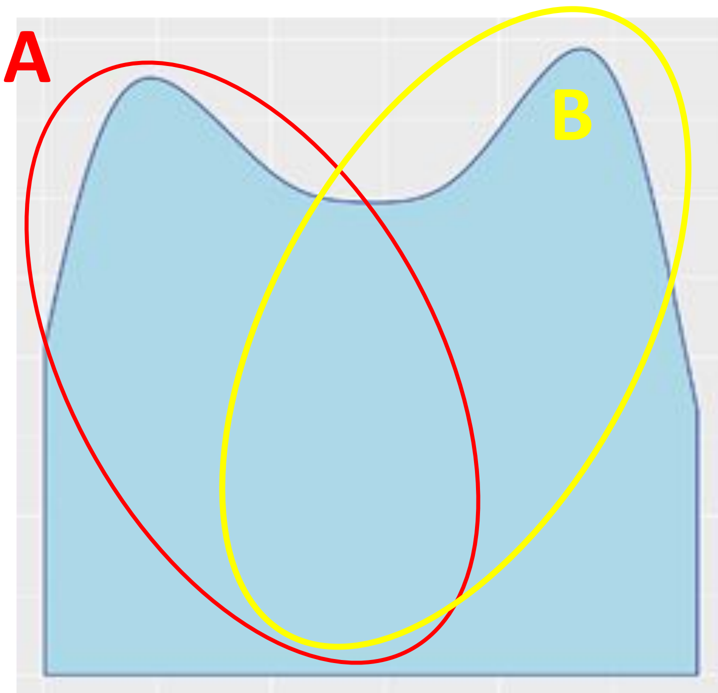
In [65]:
```
set.seed(123)
df2=valid[,81]
km.res2 <- kmeans(df2, 2, nstart = 25)
df2=as.data.frame(df2);colnames(df2)='assaults'
print(km.res2)
```

```
K-means clustering with 2 clusters of sizes 287, 283

Cluster means:
  assaults
1 133.2997
2 436.4099

Clustering vector:
  [1] 1 1 2 2 1 1 2 1 1 2 2 2 1 2 2 1 2 1 2 2 1 2 1 1 1 1 2 1 1 1 2 1 1 2 1 2 2
 [38] 2 1 1 2 2 1 1 2 2 1 1 1 2 1 2 2 1 1 2 2 1 1 2 1 2 2 2 1 2 1 2 1 2 1 2 2 2
 [75] 1 1 2 1 2 1 1 2 2 2 2 1 1 2 2 1 2 1 2 2 1 1 2 1 2 2 1 1 1 1 1 2 1 2 2 2 1
[112] 2 2 2 2 1 2 1 1 1 1 2 2 1 2 1 2 1 2 1 2 2 1 1 2 2 2 2 1 1 2 1 1 1 1 1 2 2
[149] 2 1 2 1 2 1 1 1 1 2 2 2 2 1 1 2 1 1 1 1 2 2 1 1 2 2 1 1 1 2 1 2 1 1 1 2 1
[186] 1 2 1 1 2 2 1 2 2 1 2 2 1 1 2 1 2 2 2 2 1 2 1 1 1 1 2 1 1 1 2 1 1 2 1 2 1
[223] 1 2 1 1 1 2 1 2 2 2 2 2 1 2 1 2 2 1 2 1 2 1 2 2 2 1 1 1 2 1 1 2 1 2 2 1 1
[260] 1 2 2 2 2 2 2 1 1 2 2 2 2 1 1 2 1 2 1 1 2 1 1 1 2 2 2 2 2 1 2 2 2 1 1 1 2
[297] 2 2 2 2 1 2 2 1 2 1 1 1 2 1 1 2 1 1 1 1 2 1 2 1 1 2 2 2 2 1 1 1 1 1 2 1 2
[334] 1 1 2 2 2 2 2 2 2 2 2 1 1 1 2 2 1 1 1 2 2 2 2 2 2 2 1 1 1 2 2 2 2 1 1 2 1
[371] 1 1 1 2 2 1 2 1 2 1 2 2 2 2 1 2 2 1 2 1 2 2 2 2 1 1 1 1 1 2 1 2 1 2 2 2 2
[408] 1 1 1 1 2 2 2 2 2 2 2 1 2 1 1 1 2 2 1 1 2 1 2 2 1 1 2 2 1 2 2 1 2 2 2 1 2
[445] 1 2 1 2 1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 2 2 1 1 2 2 2 1 1 1 1 2 1 2 1 2 2 2
[482] 2 1 2 1 2 1 1 2 1 2 1 1 1 2 1 1 1 1 1 2 2 2 2 1 1 1 2 2 1 1 1 1 2 1 1 1 2
[519] 2 2 1 2 2 1 1 1 2 2 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2 1 1 2 1 2 1 1 1 1 2 1 1
[556] 2 2 2 1 2 1 2 1 1 2 1 1 2 2 2
```

K-means Clustering
for Mixture Distribution

Mixture Gaussian Distributions Detected

- 데이터가 어느 클러스터에 속하는 지 결정하는 변수(**latent variable zi**)를 추가한 **hierarchical** 모형으로 **Bayesian Reg**을 시행한다면 더 나은 예측이 될 것이라 생각



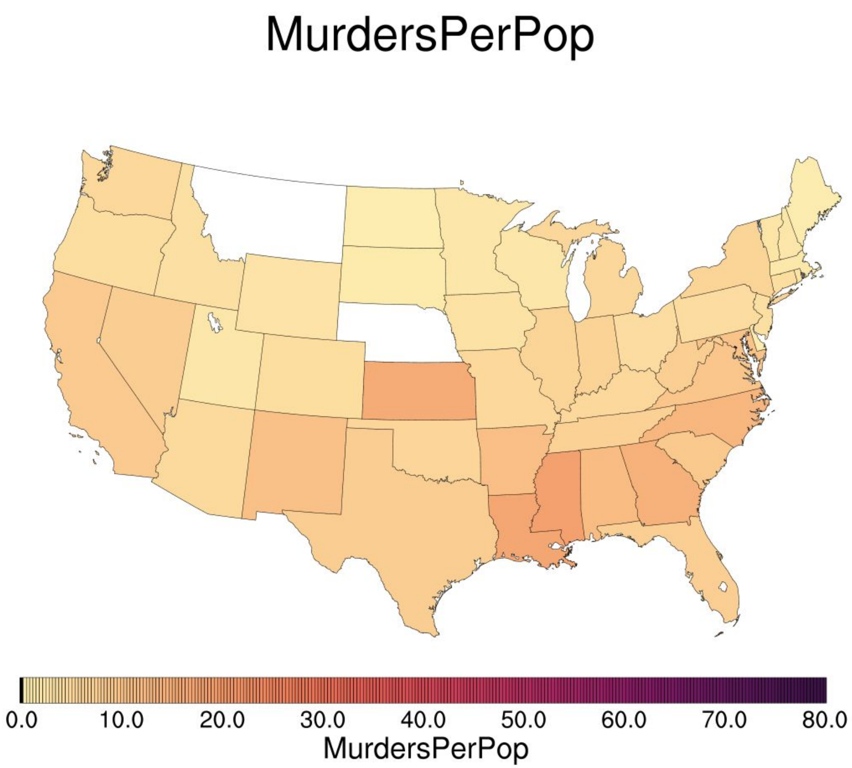
'Assaults'

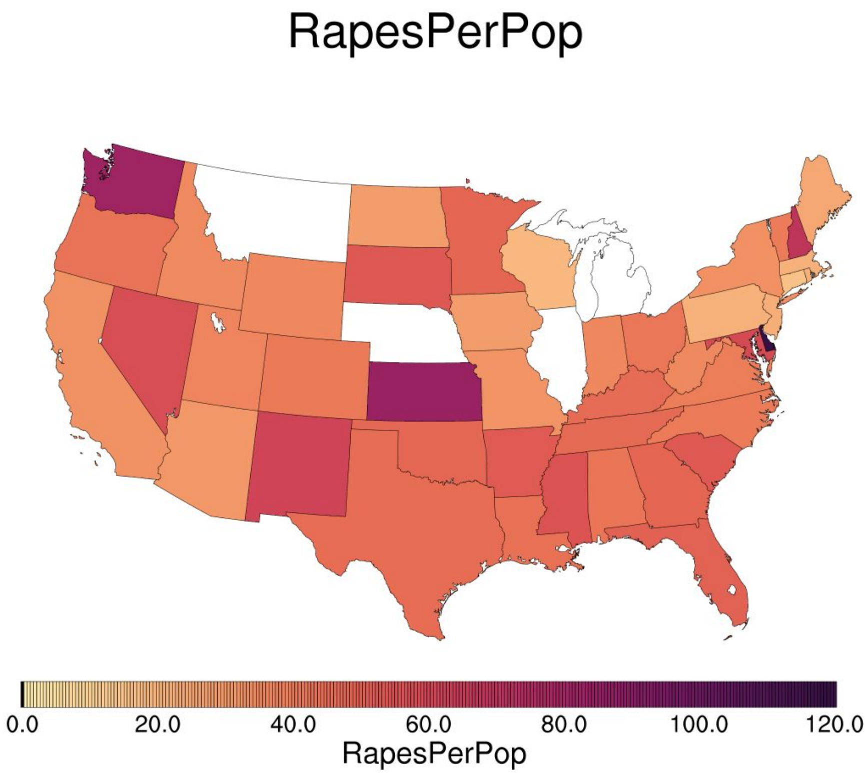
```
In [64]: valid=s.train.set[index,]
         traini=s.train.set[-index,]

In [65]: set.seed(123)
         df2=valid[,81]
         km.res2 <- kmeans(df2, 2, nstart = 25)
         df2=as.data.frame(df2);colnames(df2)='assaults'
         print(km.res2)

K-means clustering with 2 clusters of sizes 287, 283

Cluster means:
  assaults
1 133.2997
2 436.4099

Clustering vector:
  [1] 1 1 2 2 1 1 2 1 1 2 2 2 1 2 2 1 2 1 2 2 1 2 1 1 1 1 2 1 1 1 2 1 1 2 1 2 2
 [38] 2 1 1 2 2 1 1 2 2 1 1 1 2 1 2 2 1 1 2 2 1 1 2 1 2 2 2 1 2 1 2 1 2 1 2 2 2
 [75] 1 1 2 1 2 1 1 2 2 2 2 1 1 2 2 1 2 1 2 2 1 1 2 1 2 2 1 1 1 1 1 2 1 2 2 2 1
[112] 2 2 2 2 1 2 1 1 1 1 2 2 1 2 1 2 1 2 1 2 2 1 1 2 2 2 2 1 1 2 1 1 1 1 1 2 2
[149] 2 1 2 1 2 1 1 1 1 2 2 2 2 1 1 2 1 1 1 1 2 2 1 1 2 2 1 1 1 2 1 2 1 1 1 2 1
[186] 1 2 1 1 2 2 1 2 2 1 2 2 1 1 2 1 2 2 2 2 1 2 1 1 1 1 2 1 1 1 2 1 1 2 1 2 1
[223] 1 2 1 1 1 2 1 2 2 2 2 2 1 2 1 2 2 1 2 1 2 1 2 2 2 1 1 1 2 1 1 2 1 2 2 1 1
[260] 1 2 2 2 2 2 2 1 1 2 2 2 2 1 1 2 1 2 1 1 2 1 1 1 2 2 2 2 2 1 2 2 2 1 1 1 2
[297] 2 2 2 2 1 2 2 1 2 1 1 1 2 1 1 2 1 1 1 1 2 1 2 1 1 2 2 2 2 1 1 1 1 1 2 1 2
[334] 1 1 2 2 2 2 2 2 2 2 2 1 1 1 2 2 1 1 1 2 2 2 2 2 2 2 1 1 1 2 2 2 2 1 1 2 1
[371] 1 1 1 2 2 1 2 1 2 1 2 2 2 2 1 2 2 1 2 1 2 2 2 2 1 1 1 1 1 2 1 2 1 2 2 2 2
[408] 1 1 1 1 2 2 2 2 2 2 2 1 2 1 1 1 2 2 1 1 2 1 2 2 1 1 2 2 1 2 2 1 2 2 2 1 2
[445] 1 2 1 2 1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 2 2 1 1 2 2 2 1 1 1 1 2 1 2 1 2 2 2
[482] 2 1 2 1 2 1 1 2 1 2 1 1 1 2 1 1 1 1 1 2 2 2 2 1 1 1 2 2 1 1 1 1 2 1 1 1 2
[519] 2 2 1 2 2 1 1 1 2 2 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2 1 1 2 1 2 1 1 1 1 2 1 1
[556] 2 2 2 1 2 1 2 1 1 2 1 1 2 2 2
```

(K-means를 사용하여 cluster는 나누었지만 cluster를 잘 나타내는 feature은 하지 못함)

K-means Clustering
for Mixture Distribution

Mixture Gaussian Distributions Detected





Visit our Github

Our Final Code

2019 FALL FINAL Group 2 최종.ipynb

Y Outputs

finalimputation2.csv

Thank you