Summative assignment for ASML Regression

Minwoo Kim (tbdr69)

General Instructions

Please go through the R notebook below, and carry out the requested tasks. You will provide all your answers directly into this .Rmd file. Add code into the R chunks where requested. You can create new chunks where required. Where text answers are requested, please add them directly into this document, typically below the R chunks, using R Markdown syntax as adequate.

At the end, you will submit both your worked .Rmd file, and a `knitted' PDF version, through DUO.

Important: Please ensure carefully whether all chunks compile, and also check in the knitted PDF whether all R chunks did *actually* compile, and all images that you would like to produce have *actually* been generated. **An R chunk which does not compile will give zero marks, and a picture which does not exist will give zero marks, even if some parts of the required code are correct.**

Note: It is appreciated that some of the requested analyses requires running R code which is not deterministic. So, you will not have full control over the output that is finally generated in the knitted document. This is fine. It is clear that the methods under investigation carry uncertainty, which is actually part of the problem tackled in this assignment. Your analysis should, however, be robust enough so that it stays in essence correct under repeated execution of your data analysis.

Reading in data

We consider data from an industrial melter system. The melter is part of a disposal procedure, where a powder (waste material) is clad in glass. The melter vessel is continuously filled with powder, and raw glass is discretely introduced in the form of glass frit. This binary composition is heated by induction coils, positioned around the melter vessel. Resulting from this heating procedure, the glass becomes molten homogeneously (Liu et al, 2008) (https://aiche.onlinelibrary.wiley.com/doi/full/10.1002/aic.11526).

Measurements of 15 temperature sensors `temp1` , ..., `temp15` (in $^{\circ}$C), the power in four induction coils `ind1` ,..., `ind4` , the `voltage` , and the `viscosity` of the molten glass, were taken every 5 minutes. The sample size available for our analysis is n = 900.

We use the following R chunk to read the data in

```
# Clear Environment
rm(list=ls(all=T))
melter<-read.table("http://maths.dur.ac.uk/~dma0je/Data/melter.dat", header=TRUE)
```

If this has gone right, then the following code

```
is.data.frame(melter)
```

```
## [1] TRUE
```

```
dim(melter)
```

```
## [1] 900  21
```

should tell you that `melter` is a data frame of dimension 900 × 21. Otherwise something has gone wrong, and you need to start again.

To get more familiar with the data frame, please also execute

```
print('Power of induction must be over 0 because (Voltage and Current is positive)')
```

```
## [1] "Power of induction must be over 0 because (Voltage and Current is positive)"
```

```
print('And viscosity also need to be over 0 because the second law of thermodynamics requires all fluids to have
positive viscosity. Zero viscosity is observed only at near 0 Kelvin temperatures in superfluids.')
```

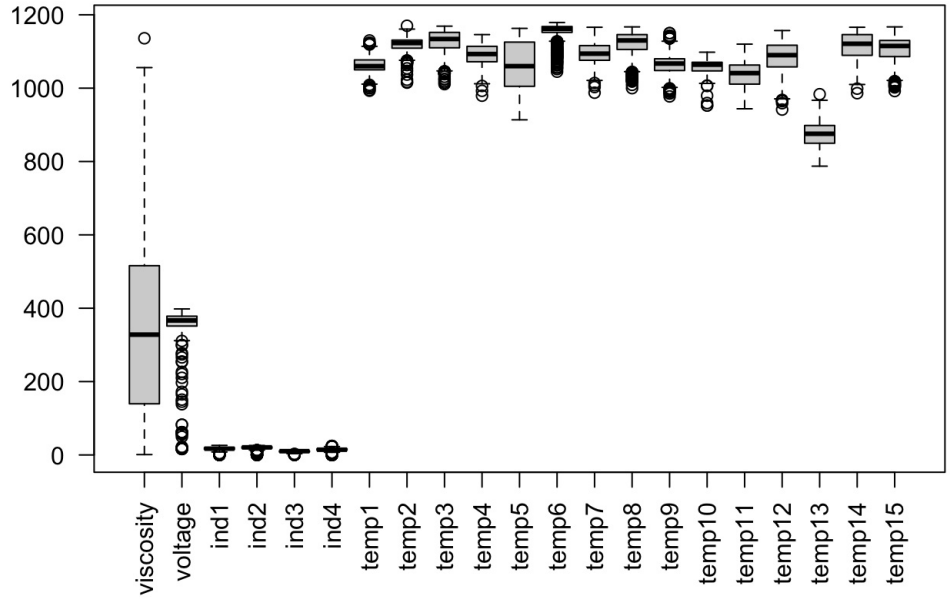
```
## [1] "And viscosity also need to be over 0 because the second law of thermodynamics requires all fluids to have
positive viscosity. Zero viscosity is observed only at near 0 Kelvin temperatures in superfluids."
```

```
# Delete unreliable values
melter<-subset(melter, !rowSums(melter < 0))
melter<-melter[melter$viscosity != 0, ]

#head(melter)
colnames(melter)
```

```
##  [1] "viscosity" "voltage"   "ind1"      "ind2"      "ind3"      "ind4"
##  [7] "temp1"     "temp2"     "temp3"     "temp4"     "temp5"     "temp6"
## [13] "temp7"     "temp8"     "temp9"     "temp10"    "temp11"    "temp12"
## [19] "temp13"    "temp14"    "temp15"
```

```
boxplot(melter, las=2)
```




```
dim(melter)
```

```
## [1] 871  21
```

Task 1: Principal component analysis (10 marks)

We consider initially only the 15 variables representing the temperature sensors. Please create a new data frame, called `Temp` , which contains only these 15 variables. Then carry out a principal component analysis. (Use your judgement on whether, for this purpose, the temperature variables require scaling or not). Produce a screeplot, and also answer the following questions: How many principal components are needed to capture 90% of the total variation? How many are needed to capture 98%?

Answer:

1. The temperature variables require scaling or not
In temperature, variables only have the same unit, so it does not need to scale.

2. How many principal components are needed to capture 90% of the total variation?
To over 90%, it needed to capture 4 Principal Component from Cumulative variance plot.

3. How many are needed to capture 98%?
For 98%, it needed to capture 7 Principal Component from Cumulative variance plot.

```
# About scailing
print('Scaling: In temperature, variables only have the same unit, so it does not need to scale.')
```

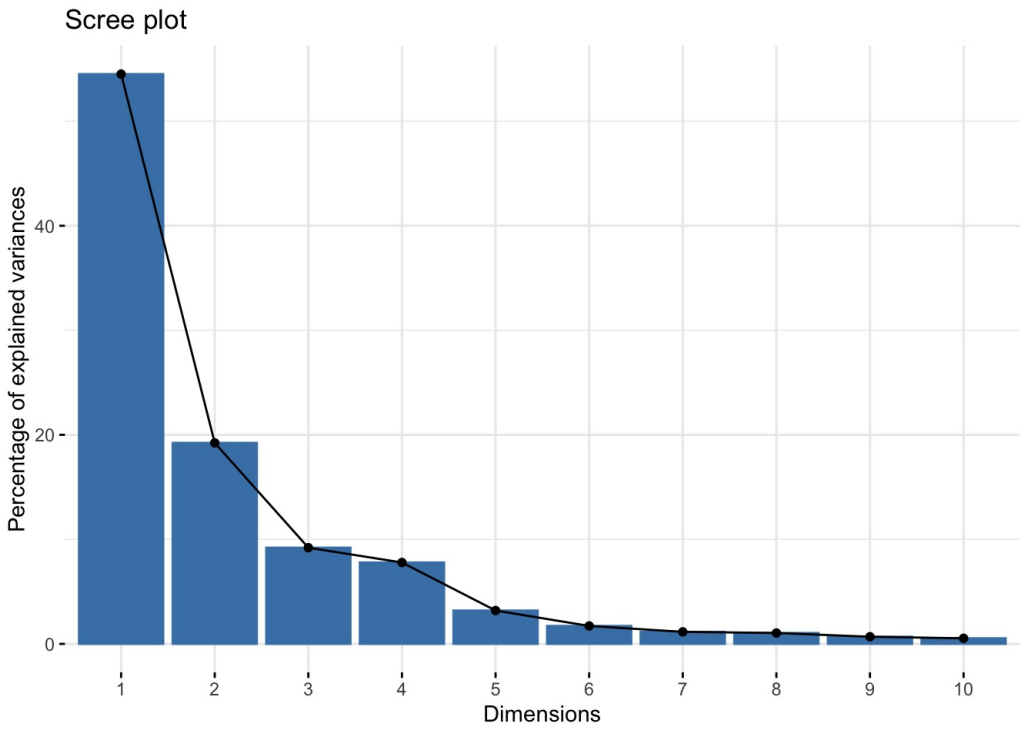
```
## [1] "Scaling: In temperature, variables only have the same unit, so it does not need to scale."
```

```
library(graphics)
library(factoextra)
```

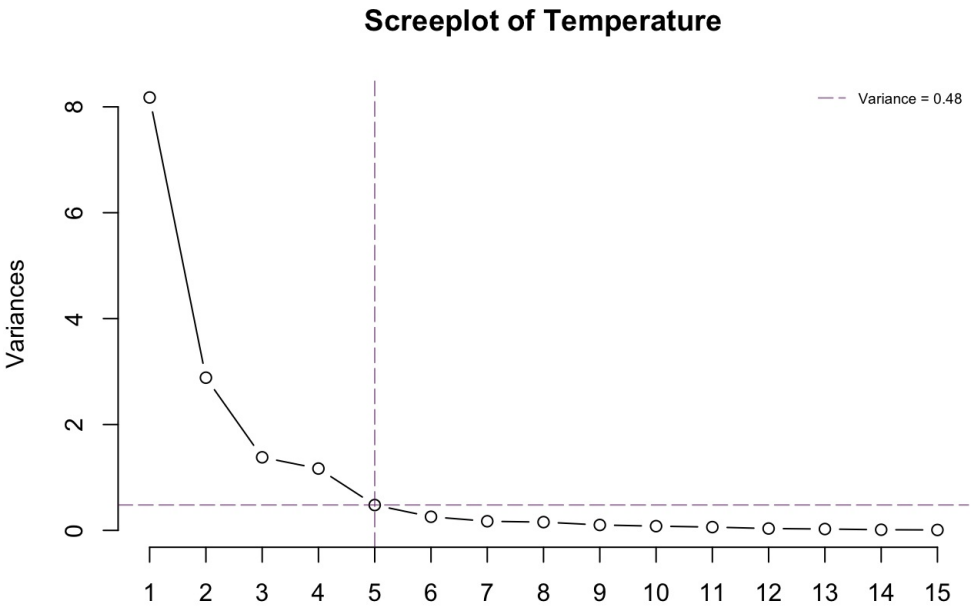
```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

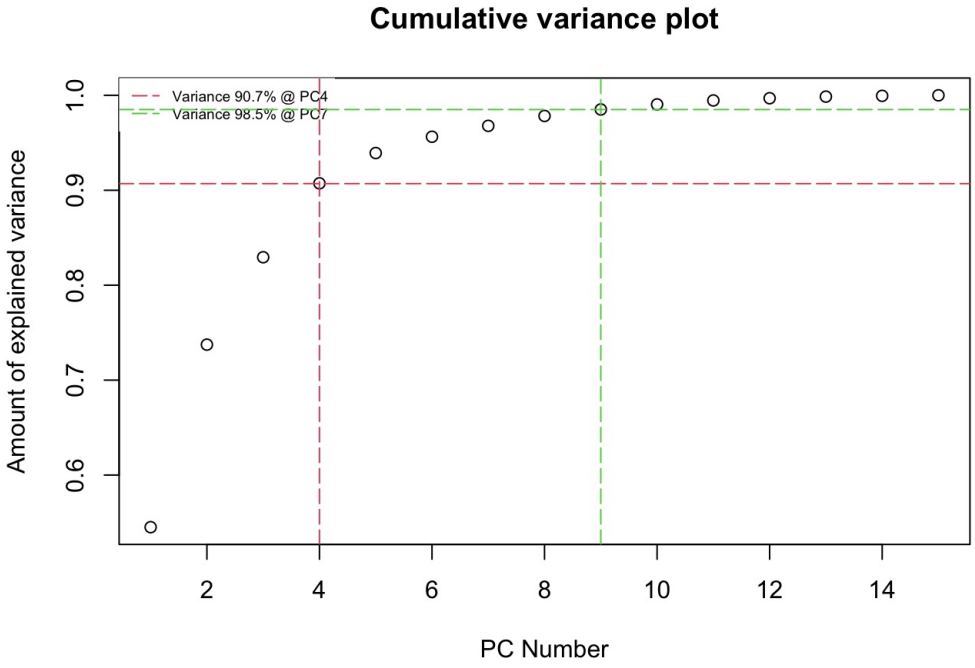
```
# Temperature variable slicing and make viscosity variable
Temp<-melter[,7:21]
Vis<-melter$viscosity
# Principal Componant Analysis with Temperature variables
Temp.comp <- prcomp(Temp, center = TRUE, scale = TRUE)
# Screeplot
fviz_eig(Temp.comp)
```




```
screeplot(Temp.comp, npcs = 15, type = "lines", main="Screeplot of Temperature")
abline(h = 0.48, col="#68246d90", lty=5)
abline(v = 5, col="#68246d90", lty=5)
legend("topright", legend=c("Variance = 0.48"), col="#68246d90", lty=5, cex=0.6, box.lty=0)
```

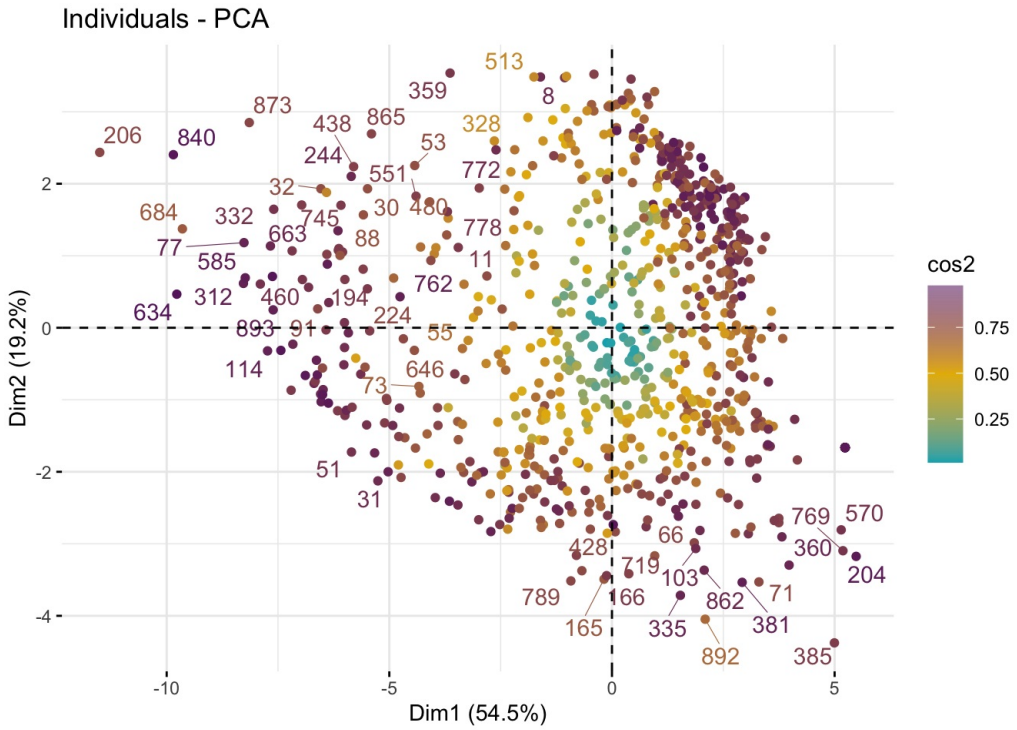



```
# Cumulative variance plot with about 90% and 98% variance
Lambda <- Temp.comp$sdev^2
cumpro <- cumsum(Lambda / sum(Lambda))
plot(cumpro[0:15], xlab = "PC Number", ylab = "Amount of explained variance", main = "Cumulative variance plot")
legend("topleft", legend=c("Variance 90.7% @ PC4",'Variance 98.5% @ PC7'), col=c(2,3), lty=5, cex=0.6, box.lty=0)
abline(h = 0.907, col=2, lty=5)
abline(v = 4, col=2, lty=5)
abline(h = 0.985, col=3, lty=5)
abline(v = 9, col=3, lty=5)
```



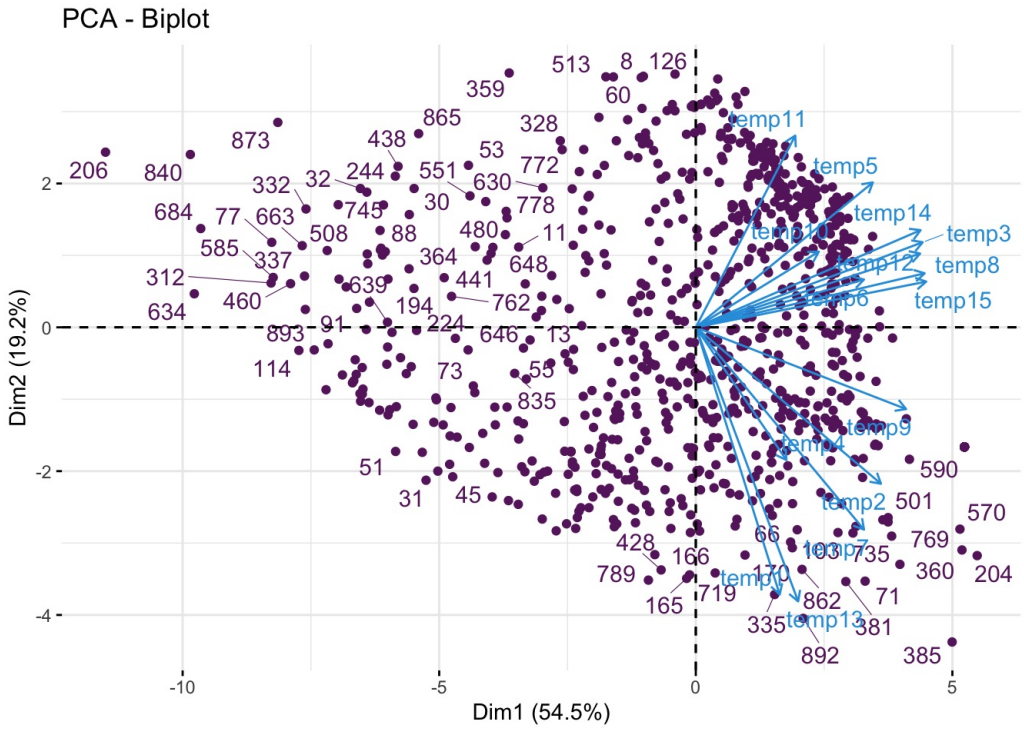

```
# PCA plot
fviz_pca_ind(Temp.comp, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#68246d90"), repel = TRUE)
```

```
## Warning: ggrepel: 815 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
fviz_pca_biplot(Temp.comp, repel = TRUE, col.var = "#2E9FDF", col.ind = "#68246d90")
```

```
## Warning: ggrepel: 799 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
print('How many principal components are needed to capture 90% of the total variation? ')
```

```
## [1] "How many principal components are needed to capture 90% of the total variation? "
```

```
print('To over 90% it needed to capture 4 Principal Component')
```

```
## [1] "To over 90% it needed to capture 4 Principal Component"
```

```
print('How many are needed to capture 98%?')
```

```
## [1] "How many are needed to capture 98%?"
```

```
print('For 98% it needed to capture 7 Principal Componant')
```

```
## [1] "For 98% it needed to capture 7 Principal Componant"
```

Task 2: Multiple linear regression (20 marks)

We consider from now on, and for the remainder of this assignment, `viscosity` as the response variable.

Fit a linear regression model, with `viscosity` as response variable, and all other variables as predictors, and produce the `summary` output of the fitted model. In this task, we are mainly interested in the standard errors of the estimated coefficients. Create a vector, with name `melter.fit.sd` , which contains the standard errors of all estimated coefficients, except the intercept. (So, this vector should have length 20). Then produce a `barplot` of these standard errors (where the height of each bar indicates the value of the standard error of the respective coefficients). Please use blue color to fill the bars of the barplot.

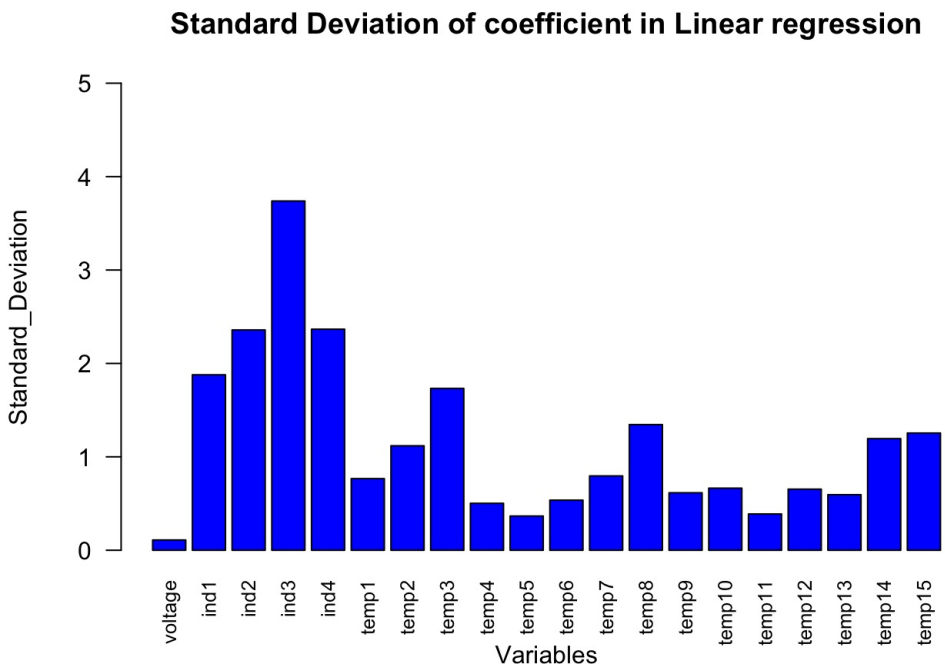
Answer:

In following all methods' Standard Error (SE) was calculated as Standard Deviation (SD). Although some of the methods have sampling procedures, this needs to be compared with other methods, so the Standard Error of Mean would not fit and could count as one sampling in this one dataset case. So from here, Standard Error = Standard Deviation.

```
# Fit multiple Linear regression model
vis.model<-lm(viscosity~., data=melter)
# View results of model
vis.summary<-summary(vis.model)
print(vis.summary$coefficients)
```

```
##                 Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept) -763.6586134 718.5596949 -1.0627629 2.881913e-01
## voltage        0.3315331   0.1097443  3.0209594 2.595222e-03
## ind1          -1.4089115   1.8785391 -0.7500038 4.534600e-01
## ind2          11.7521826   2.3583933  4.9831308 7.581216e-07
## ind3          -3.7451252   3.7387233 -1.0017123 3.167676e-01
## ind4           4.1634585   2.3665301  1.7593093 7.888472e-02
## temp1          3.6479860   0.7669114  4.7567240 2.311019e-06
## temp2         -5.9364226   1.1184090 -5.3079176 1.415986e-07
## temp3         -0.4317721   1.7327790 -0.2491790 8.032825e-01
## temp4         -0.8963324   0.5024886 -1.7837866 7.481502e-02
## temp5          0.5551032   0.3662170  1.5157766 1.299476e-01
## temp6          1.3486996   0.5361414  2.5155671 1.206744e-02
## temp7         -0.2749364   0.7957883 -0.3454894 7.298120e-01
## temp8          7.3243936   1.3452904  5.4444704 6.803026e-08
## temp9         -2.4142647   0.6152153 -3.9242599 9.402657e-05
## temp10         0.6317495   0.6645293  0.9506721 3.420411e-01
## temp11        -0.4593230   0.3882162 -1.1831629 2.370754e-01
## temp12         1.3040200   0.6538825  1.9942726 4.644131e-02
## temp13        -0.2269712   0.5949589 -0.3814906 7.029346e-01
## temp14         3.1748972   1.1950919  2.6566134 8.040974e-03
## temp15        -6.7366619   1.2545255 -5.3698883 1.017322e-07
```

```
# Get the standard error of Linear regression
melter.fit.sd<-vis.summary$coefficients[,2][2:21]
# Name of variables
xname<-colnames(melter[2:21])
# Bar Plot
barplot(melter.fit.sd, col='blue',names.arg=c(xname),ylim=c(0,5),main='Standard Deviation of coefficient in Linea
r regression', ylab="Standard_Deviation", xlab="Variables",las=2, cex.names = 0.8)
```



Now repeat this analysis, but this time using a Bayesian linear regression. Use adequate methodology to fit the Bayesian version of the linear model considered above. It is your choice whether you would like to employ ready-to-use R functions which carry out this task for you, or whether you would like to implement this procedure from scratch, for instance using `jags`.

In either case, you need to be able to extract posterior draws of the estimated parameters from the fitted object, and compute their standard deviations. Please save these standard deviations, again excluding that one for the intercept, into a vector `melter.bayes.sd`. Produce now a barplot which displays both of `melter.fit.sd` and `melter.bayes.sd` in one plot, and allows a direct comparison of the frequentist and Bayesian standard errors (by having the corresponding bars for both methods directly side-by-side, with the Bayesian ones in red color). The barplot should be equipped with suitable labels and legends to enable good readability.

Comment on the outcome.

Answer:

```
library(rstanarm)
```

```
## Loading required package: Rcpp
```

```
## This is rstanarm version 2.21.1
```

```
## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
```

```
## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.
```

```
## - For execution on a local, multicore CPU with excess RAM we recommend calling
```

```
##   options(mc.cores = parallel::detectCores())
```

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
# Linear bayesian regression function in  rstanarm
vis.bayes <- stan_glm(viscosity~., data=melter, seed=1, refresh = 0)
```

```
library(ggplot2)
library(reshape2)
# Summary of Bayesian linear regression
vis.bayes.summary<-as.data.frame(summary(vis.bayes,digits = 8))
# Standard deviation of Bayesian linear regression
melter.functionbayes.sd<-vis.bayes.summary$sd[2:21]
# Slice of Melter
melter.re<-melter[,2:21]
```

```
# Need to keep the standard deviation's for later unscaling
Stan <- apply(melter.re, 2, sd)
X <- model.matrix(vis.model)
bayes.string <- "model{
  for(i in 1:N){
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- beta0+beta%*%X[i,]
  }
  # Prior distribution on mean
    beta0 ~ dnorm(0, 0.0001);
    for (j in 1:20){
      beta[j]~  dnorm(0, 0.0001)
    }
    tau  ~ dgamma(0.01, 0.01)
    sigma<- 1/sqrt(tau)
}"
# Do Jags Baysian linear regression and get result
bayes.model <- jags.model(textConnection(bayes.string), data = list(X=scale(melter.re), y=Vis,N=dim(as.matrix(mel
ter.re))[1]))
```

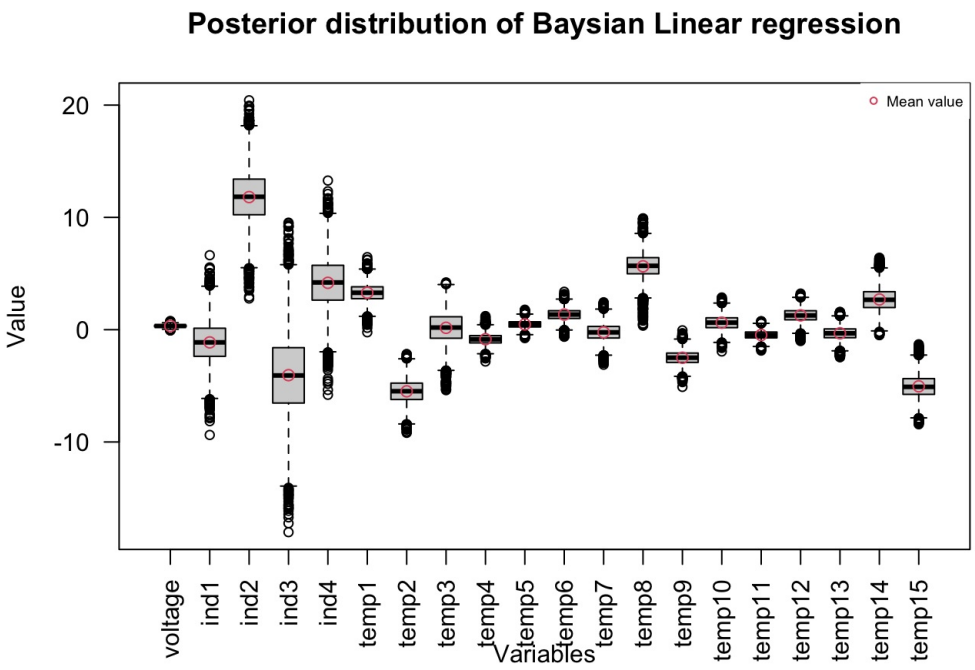
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 871
##    Unobserved stochastic nodes: 22
##    Total graph size: 20870
##
## Initializing model
```

```
update(bayes.model,10000)
summary(bayes.model)
```

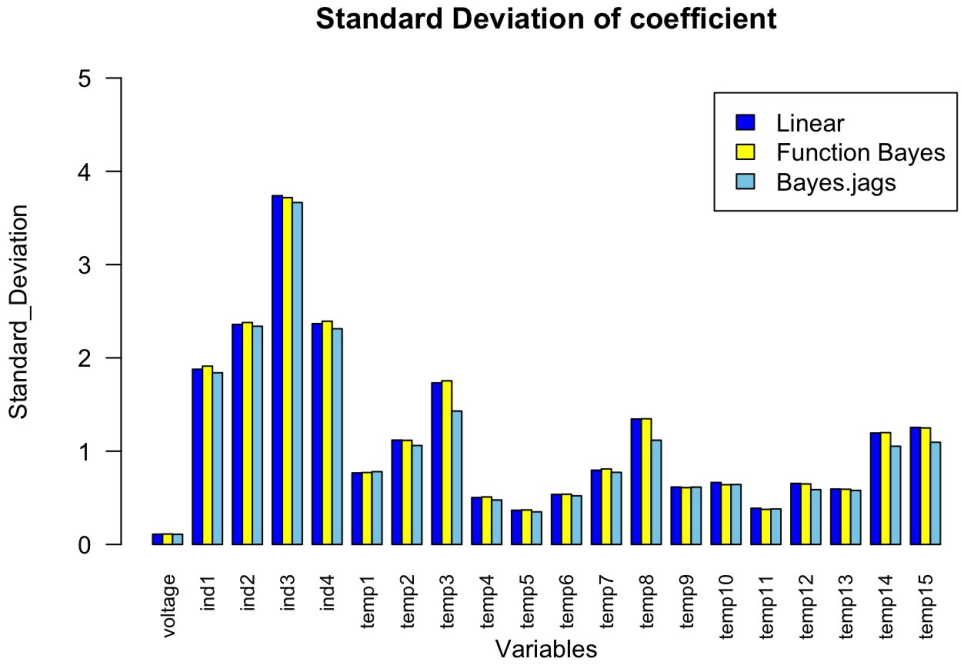
```
##           Length Class  Mode
## ptr       1      -none- function
## data      1      -none- function
## model     1      -none- function
## state     1      -none- function
## nchain    1      -none- function
## iter      1      -none- function
## sync      1      -none- function
## recompile 1      -none- function
```

```
posterior.samples = coda.samples(bayes.model, c("beta0", "beta", "sigma"), 10000)[[1]]
# Get Standard deviation of Jags
melter.bayes.sd <- round(apply(t(posterior.samples[,1:20])/Stan, 1,sd), digits=4)
```

```
# Plot Boxplot about Posterior distribution
coe.jags<- rowMeans(t(posterior.samples[,1:20])/Stan)
boxplot(as.data.frame(t(t(posterior.samples[,1:20])/Stan)),names=xname,main='Posterior distribution of Baysian Li
near regression', ylab="Value", xlab="Variables",las=2, cex = 0.8)
points(as.data.frame(coe.jags), col=2)
legend("topright", legend=c("Mean value"), col=c(2), pch=1, cex=0.6, box.lty=0)
```




```
# Making dataset of standard deviation
df <- as.data.frame(cbind(melter.fit.sd, melter.functionbayes.sd, melter.bayes.sd))
val<-matrix(0, nrow = 3, ncol = 20)
# An empty matrix is a necessary requirement prior to copying data
val[1,] <- df$melter.fit.sd
val[2,] <- df$melter.functionbayes.sd
val[3,] <- df$melter.bayes.sd
# Plot barplot to compare
barplot(val, names.arg = xname, beside = TRUE,ylim=c(0,5), col = c('blue','yellow','skyblue'), legend.text = c("L
inear", "Function Bayes",'Bayes.jags'),main='Standard Deviation of coefficient', ylab="Standard_Deviation", xlab=
"Variables",las=2, cex.names = 0.8)
```




```
# Conclusion
print('From the barplot, Jags are a little more accurate than functional Bayesian linear regression. So after tha
t, using Jags Bayesian linear regression version.')
```

```
## [1] "From the barplot, Jags are a little more accurate than functional Bayesian linear regression. So after th
at, using Jags Bayesian linear regression version."
```

```
print('From the barplot, Linear regression and functional Bayes linear regression is not much different. But if i
t uses the Jags Bayesian linear regression, the standard deviation of coefficients are slightly less than other m
ethods.
It seems like calculating posterior with prior manually makes less loss than functional Bayesian linear regressio
n.')
```

```
## [1] "From the barplot, Linear regression and functional Bayes linear regression is not much different. But if
it uses the Jags Bayesian linear regression, the standard deviation of coefficients are slightly less than other
methods.\nIt seems like calculating posterior with prior manually makes less loss than functional Bayesian linear
regression."
```

Task 3: The Lasso (20 marks)

We would like to reduce the dimension of the currently 20-dimensional space of predictors. We employ the LASSO to carry out this task. Carry out this analysis, which should feature the following elements:

- the trace plot of the fitted coefficients, as a function of log(λ), with λ denoting the penalty parameter;
- a graphical illustration of the cross-validation to find λ;
- the chosen value of λ according to the `1-se` criterion, and a brief explanation of what this criterion actually does;
- the fitted coefficients according to this choice of λ.

Answer:

- the chosen value of λ according to the `1-se` criterion, and a brief explanation of what this criterion actually does :

  Lambda.1se offers the most regularized model such that error is within one standard error of the minimum. This is fundamental for a much more conservative approach to selection. The number of nonzero coefficients is shown along the top of the plot. Minimum Lambda is the value of lambda that gives minimum mean cross-validated error.

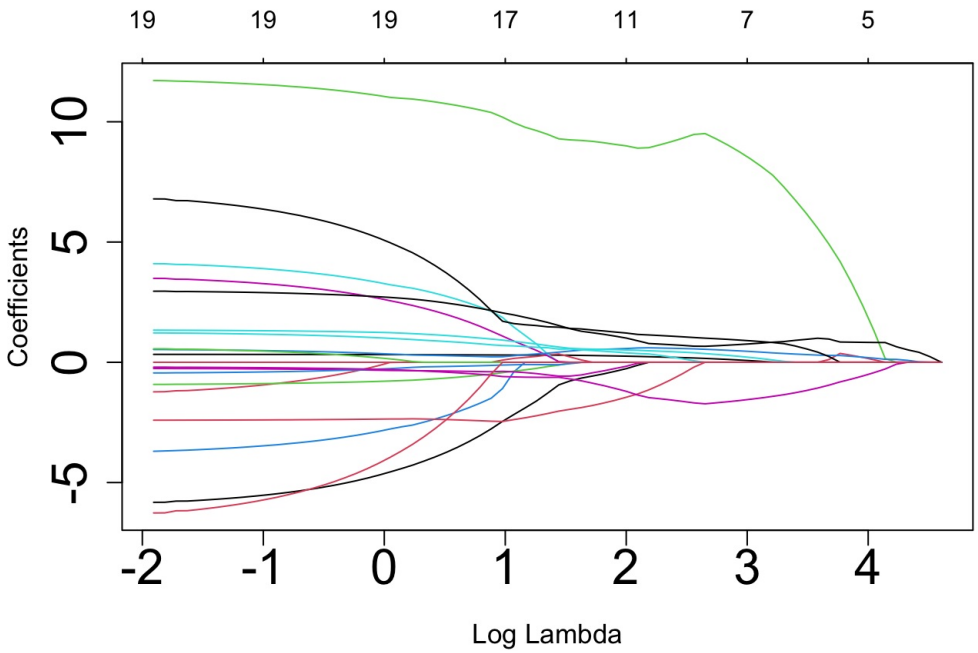
  In this dataset, for model selection, lambda.1se was selected.

```
library(glmnet)
```

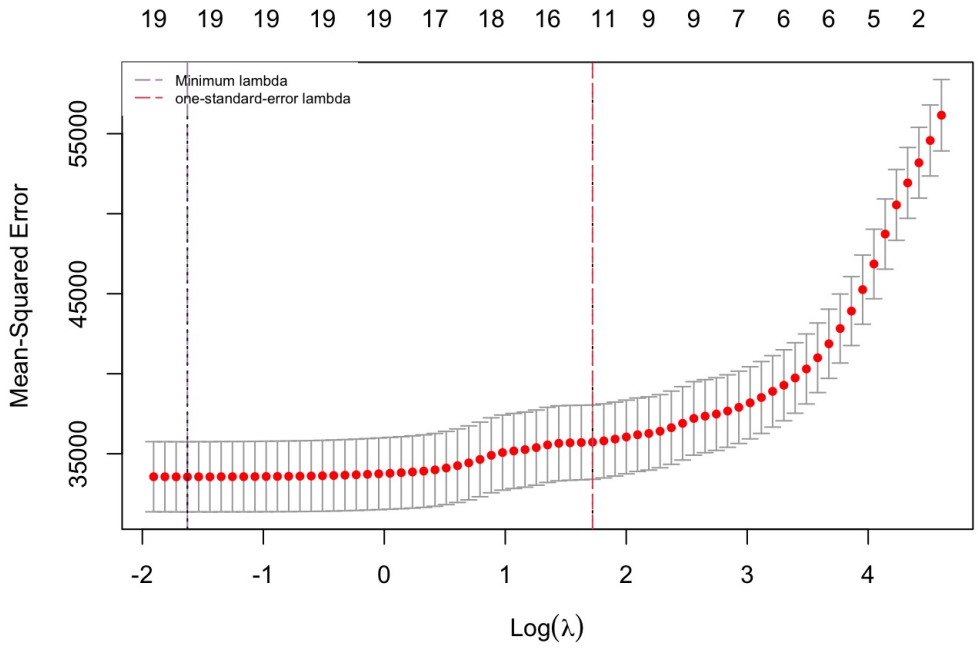
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

```
set.seed(1)
# Fit the model viscosity-variables to generalized linear model to find trace of coefficients
model <- glmnet(melter[,2:21], Vis, alpha = 1, family="gaussian")
plot(model, xvar="lambda", cex.axis=2, cex.lab=1.1, cex=1.1)
```




```
# Perform cross-validation to find lambda values
cv.melter<-cv.glmnet(as.matrix(melter[,2:21]), Vis, alpha=1, nfold=100)
# Find optimal lambdas value
best.lambda <- cv.melter$lambda.min
se.lambda <- cv.melter$lambda.1se
# Plot cross-validation plot to find lambda
par(mfrow=c(1,1))
plot(cv.melter)
legend("topleft", legend=c("Minimum lambda",'one-standard-error lambda'), col=c('#68246d90',2), lty=5, cex=0.6, b
ox.lty=0)
abline(v = log(best.lambda), col="#68246d90", lty=5)
abline(v = log(se.lambda), col=2, lty=5)
```



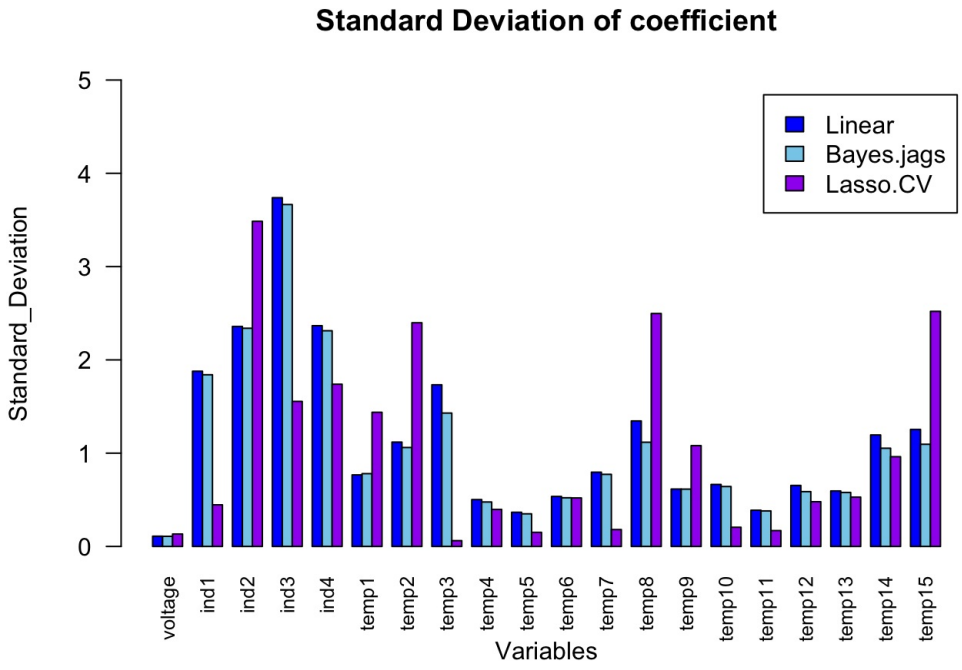
```
# Find coefficients of best model with one-standard-error lambda
best.model <- glmnet(melter[,2:21], Vis, lambda=se.lambda, alpha = 1, family="gaussian")
cv.lasso.summary<-summary(cv.melter)
# Clear NaN and Inf value in coefficients and find variables index
coe<-as.matrix(coef(best.model))
coe[is.nan(coe)] <- 0
coe[is.infinite(coe)] <- 0
coe.val<-coe[which(coe[,1] != 0)]
coe.ind<-rownames(coe)[which(coe[,1] != 0)]
# Question and Conclustion
print('The chosen value of lambda according to the 1-se criterion, and a brief explanation of what this criterion
actually does')
```

```
## [1] "The chosen value of lambda according to the 1-se criterion, and a brief explanation of what this criterio
n actually does"
```

```
print('Lambda.1se offers the most regularized model such that error is within one standard error of the minimum.
This is fundamental for a much more conservative approach to selection. The number of nonzero coefficients is sho
wn along the top of the plot. Minimum Lambda is the value of lambda that gives minimum mean cross-validated error
.
In this dataset, for model selection, lambda.1se was selected.')
```

```
## [1] "Lambda.1se offers the most regularized model such that error is within one standard error of the minimum.
This is fundamental for a much more conservative approach to selection. The number of nonzero coefficients is sho
wn along the top of the plot. Minimum Lambda is the value of lambda that gives minimum mean cross-validated error
.\nIn this dataset, for model selection, lambda.1se was selected."
```

```
# Get the all coefficient data and find standard deviation
meldf<-as.data.frame(t(as.matrix(cv.melter$glmnet.fit$beta)))
melter.lasso.cv.sd<-apply(meldf,2,sd)
# Make dataset for barplot
df <- as.data.frame(cbind(melter.fit.sd, melter.bayes.sd, melter.lasso.cv.sd))
val<-matrix(0, nrow = 3, ncol = 20)
# An empty matrix is a necessary requirement prior to copying data
val[1,] <- df$melter.fit.sd
val[2,] <- df$melter.bayes.sd
val[3,] <- df$melter.lasso.cv.sd
#An empty matrix is a necessary requirement prior to copying data
barplot(val, names.arg = xname, beside = TRUE,ylim=c(0,5), col = c('blue','skyblue','purple'), legend.text = c("L
inear",'Bayes.jags','Lasso.CV'),main='Standard Deviation of coefficient', ylab="Standard_Deviation", xlab="Variab
les",las=2, cex.names = 0.8)
```




```
# Conclustion 2
# From the Standard Deviation result CV.Lasso does not have merit, so after that delete
```

Next, carry out a Bayesian analysis of the lasso. Visualize the full posterior distributions of all coefficients (except the intercept) in terms of boxplots, and also visualize the resulting standard errors of the coefficients, again using a barplot (with red bars).

Give an interpretation of the results, especially in terms of the evidence that this analysis gives in terms of inclusion/non-inclusion of certain variables.

Answer:

Lasso have a variable selection because Lasso regression converges to zero at a certain point in time, and several variables can be non-selective, and there are L1 penalty, so Lasso allows the absolute value of the regression coefficient to come within a certain level.

In Bayesian, Lasso regression has MCMC iteration. Laplace prior will produce an identical reducing effect to the L1 penalty during the iteration (blasso function in R has reversible jump that affect to variable selection). In this data, a variable selection criterion is the number of counting(by a person in the general case, but posterior probability could use). If the count is high, the posterior probability is also higher than other posterior probability. In this case, select 9000 counts for the selection.

The normal Lasso will probably be more efficient (quick) to compute than the Bayesian Lasso.

```
library(monomvn)
```

```
## Loading required package: pls
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:rstanarm':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
## Loading required package: lars
```

```
## Loaded lars 1.2
```

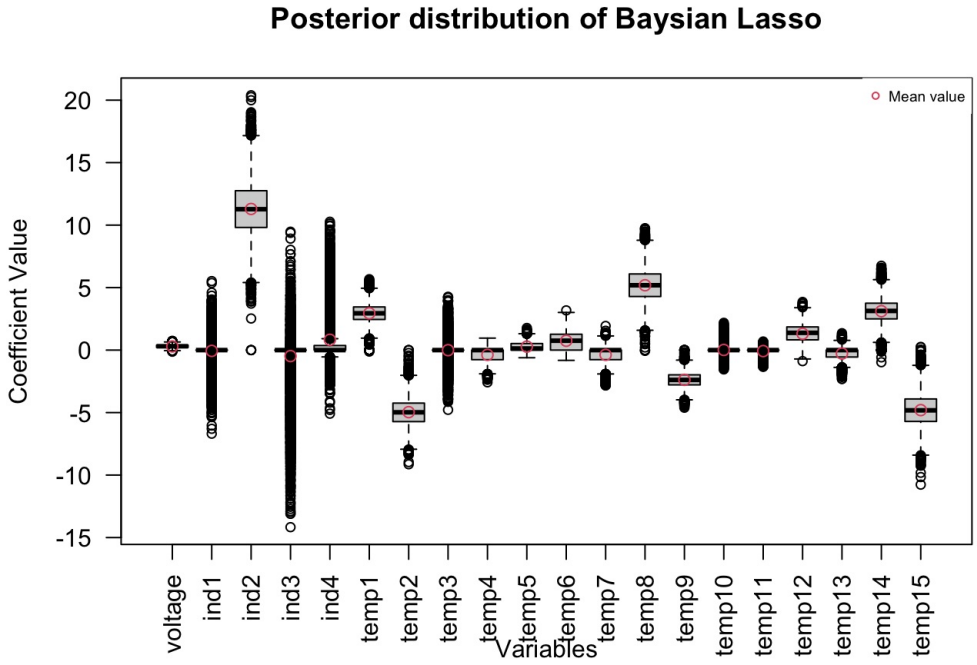
```
## Loading required package: MASS
```

```
set.seed(1)
# Fit the model viscosity and variables to Bayesian Lasso
model2 <- blasso(melter[,2:21], Vis, T=10000)
```

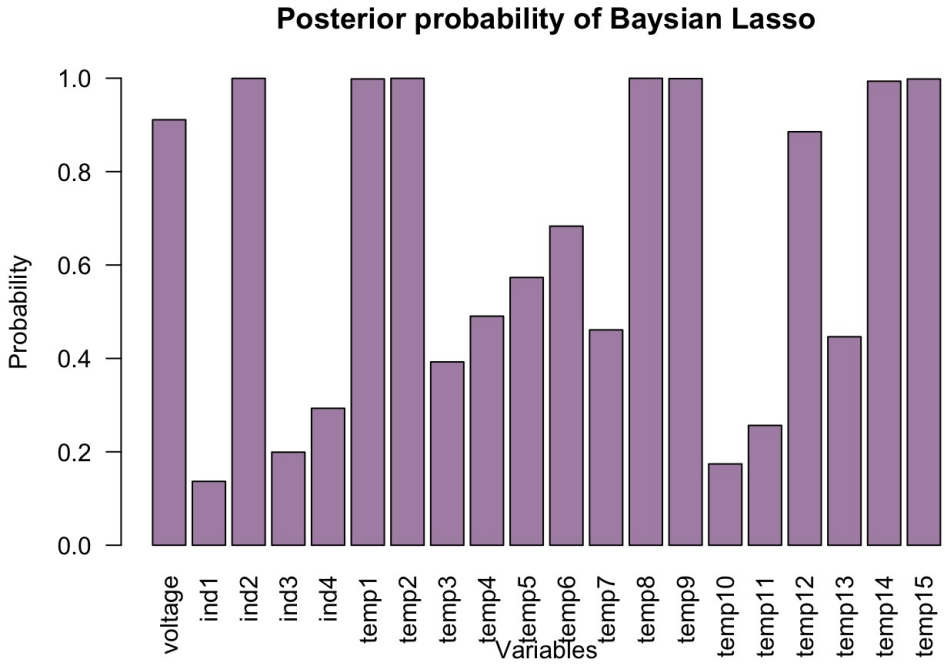
```
## t=100, m=15
## t=200, m=13
## t=300, m=15
## t=400, m=13
## t=500, m=12
## t=600, m=15
## t=700, m=13
## t=800, m=15
## t=900, m=13
## t=1000, m=15
## t=1100, m=13
## t=1200, m=11
## t=1300, m=14
## t=1400, m=12
## t=1500, m=15
## t=1600, m=13
## t=1700, m=13
## t=1800, m=12
## t=1900, m=15
## t=2000, m=13
## t=2100, m=11
## t=2200, m=13
## t=2300, m=14
## t=2400, m=12
## t=2500, m=15
## t=2600, m=11
## t=2700, m=13
## t=2800, m=14
## t=2900, m=14
## t=3000, m=13
## t=3100, m=12
## t=3200, m=11
## t=3300, m=13
```

```
## t=3400, m=11
## t=3500, m=14
## t=3600, m=15
## t=3700, m=12
## t=3800, m=14
## t=3900, m=12
## t=4000, m=11
## t=4100, m=13
## t=4200, m=13
## t=4300, m=12
## t=4400, m=12
## t=4500, m=12
## t=4600, m=14
## t=4700, m=14
## t=4800, m=12
## t=4900, m=14
## t=5000, m=13
## t=5100, m=13
## t=5200, m=13
## t=5300, m=14
## t=5400, m=10
## t=5500, m=13
## t=5600, m=10
## t=5700, m=15
## t=5800, m=13
## t=5900, m=13
## t=6000, m=13
## t=6100, m=12
## t=6200, m=11
## t=6300, m=11
## t=6400, m=14
## t=6500, m=12
## t=6600, m=13
## t=6700, m=12
## t=6800, m=12
## t=6900, m=13
## t=7000, m=14
## t=7100, m=13
## t=7200, m=11
## t=7300, m=12
## t=7400, m=13
## t=7500, m=13
## t=7600, m=12
## t=7700, m=11
## t=7800, m=12
## t=7900, m=12
## t=8000, m=12
## t=8100, m=12
## t=8200, m=14
## t=8300, m=11
## t=8400, m=13
## t=8500, m=13
## t=8600, m=11
## t=8700, m=12
## t=8800, m=12
## t=8900, m=15
## t=9000, m=12
## t=9100, m=11
## t=9200, m=12
## t=9300, m=14
## t=9400, m=13
## t=9500, m=15
## t=9600, m=11
## t=9700, m=11
## t=9800, m=15
## t=9900, m=14
```

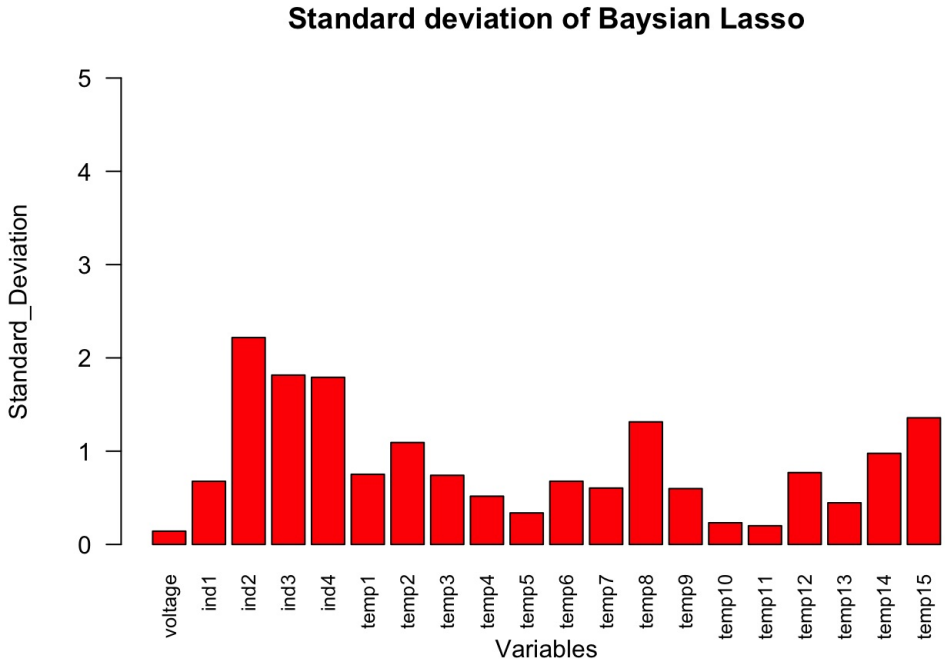
```
post <- summary(model2)$bn0
# Plot the Posterior distribution and barplot of coefficients
boxplot(model2$beta, names=xname,main='Posterior distribution of Baysian Lasso', ylab="Coefficient Value", xlab="
Variables",las=2, cex = 0.8)
points(apply(model2$beta,2,mean),col=2)
legend("topright", legend=c("Mean value"), col=c(2), pch=1, cex=0.6, box.lty=0)
```



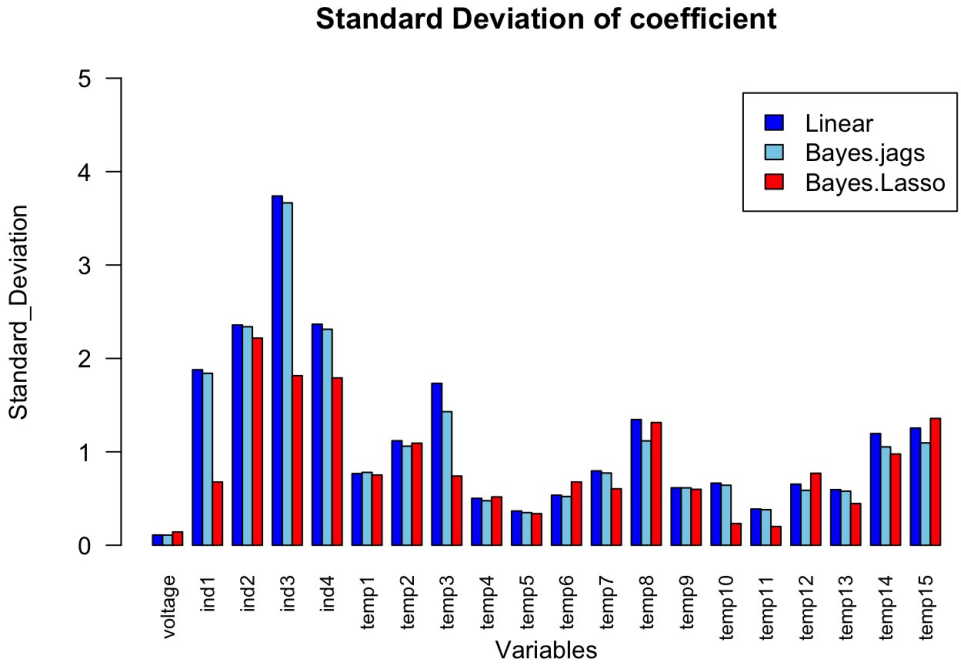

```
barplot(t(post),names.arg=xname,col=c('#68246d90'),ylim=c(0,1), main='Posterior probability of Baysian Lasso', yl
ab="Probability", xlab="Variables",las=2)
```




```
# Find standard deviation and plot barplot
bay.sd<-apply(model2$beta,2,sd)
barplot(bay.sd, names.arg=xname, col='red',ylim=c(0,5),main='Standard deviation of Baysian Lasso', ylab="Standard
_Deviation", xlab="Variables",las=2, cex.names = 0.8)
```




```
# Clear NaN and Inf value in coefficients and find gene name index
coe2<-as.matrix(model2$beta)
# Count number of coefficients which selected during MCMC
num<-as.data.frame(apply(coe2, 2, function(c)sum(c!=0)))
rownames(num)<-rownames(coe)[2:21]
colnames(num)<-c('Count')
coe2[is.nan(coe2)] <- 0
coe2[is.infinite(coe2)] <- 0
coe.val2<-coe2[which(coe2[,1] != 0)]
coe.ind2<-rownames(coe2)[which(coe2[,1] != 0)]
# Make dataset for barplot
df <- as.data.frame(cbind(melter.fit.sd, melter.bayes.sd, bay.sd))
val<- matrix(0, nrow = 3, ncol = 20)
# An empty matrix is a necessary requirement prior to copying data
val[1,] <- df$melter.fit.sd
val[2,] <- df$melter.bayes.sd
val[3,] <- df$bay.sd
barplot(val, names.arg = xname, beside = TRUE,ylim=c(0,5), col = c('blue','skyblue','red'), legend.text = c("Line
ar",'Bayes.jags','Bayes.Lasso'),main='Standard Deviation of coefficient', ylab="Standard_Deviation", xlab="Variab
les",las=2, cex.names = 0.8)
```



```
# Auxiliary data for conclusion
coe.val2<-coe2[which(coe2[,1] != 0)]
coe.ind2<-rownames(coe2)[which(coe2[,1] != 0)]
coe.val2<-num[which(num$Count>=7000),1]
coe.ind2<-rownames(num)[which(num$Count>=7000)]
print('Variable selection of original Lasso regression')
```

```
## [1] "Variable selection of original Lasso regression"
```

```
print(coe.ind[2:12])
```

```
##  [1] "voltage" "ind1"    "ind2"    "temp2"   "temp5"   "temp6"   "temp7"
##  [8] "temp8"   "temp9"   "temp12"  "temp13"
```

```
print('Variable selection of Bayesian Lasso regression and number of selected in MCMC iteration')
```

```
## [1] "Variable selection of Bayesian Lasso regression and number of selected in MCMC iteration"
```

```
print(coe.ind2)
```

```
## [1] "voltage" "ind2"    "temp1"   "temp2"   "temp8"   "temp9"   "temp12"
## [8] "temp14"  "temp15"
```

```
print(as.data.frame(as.matrix(rbind(xname,post,num$Count))))
```

```
##           b.1    b.2    b.3    b.4    b.5    b.6    b.7    b.8    b.9   b.10
## xname voltage   ind1   ind2   ind3   ind4  temp1  temp2  temp3  temp4  temp5
## post    0.911 0.1367 0.9996 0.1993 0.2932 0.9984 0.9997 0.3926 0.4905 0.5733
##          9110   1367   9996   1993   2932   9984   9997   3926   4905   5733
##         b.11   b.12   b.13   b.14   b.15   b.16   b.17   b.18   b.19   b.20
## xname  temp6  temp7  temp8  temp9 temp10 temp11 temp12 temp13 temp14 temp15
## post  0.6832 0.4611 0.9998 0.9991 0.1741 0.2566 0.8854 0.4464 0.9936 0.9983
##         6832   4611   9998   9991   1741   2566   8854   4464   9936   9983
```

```
# Question and Conclustion
print('Give an interpretation of the results, especially in terms of the evidence that this analysis gives in ter
ms of inclusion/non-inclusion of certain variables.')
```

```
## [1] "Give an interpretation of the results, especially in terms of the evidence that this analysis gives in te
rms of inclusion/non-inclusion of certain variables."
```

```
print('Lasso have a variable selection because Lasso regression converges to zero at a certain point in time, and
several variables can be non-selective, and there are L1 penalty, so Lasso allows the absolute value of the regre
ssion coefficient to come within a certain level.')
```

```
## [1] "Lasso have a variable selection because Lasso regression converges to zero at a certain point in time, an
d several variables can be non-selective, and there are L1 penalty, so Lasso allows the absolute value of the reg
ression coefficient to come within a certain level."
```

```
print('In Bayesian, Lasso regression has MCMC iteration. Laplace prior will produce an identical reducing effect
to the L1 penalty during the iteration (blasso function in R has reversible jump that affect to variable selectio
n). In this data, a variable selection criterion is the number of counting( by a person in the general case, but
posterior probability could use ). If the count is high, the posterior probability is also higher than other post
erior probability. In this case, select 9000 counts for the selection.
The normal Lasso will probably be more efficient (quick) to compute than the Bayesian Lasso.')
```

```
## [1] "In Bayesian, Lasso regression has MCMC iteration. Laplace prior will produce an identical reducing effect
to the L1 penalty during the iteration (blasso function in R has reversible jump that affect to variable selectio
n). In this data, a variable selection criterion is the number of counting( by a person in the general case, but
posterior probability could use ). If the count is high, the posterior probability is also higher than other post
erior probability. In this case, select 9000 counts for the selection.\nThe normal Lasso will probably be more ef
ficient (quick) to compute than the Bayesian Lasso."
```

Task 4: Bootstrap (20 marks)

A second possibility to assess uncertainty of any estimator is the Bootstrap. Implement a nonparametric bootstrap procedure to assess the uncertainty of your frequentist lasso fit from Task 3.

Produce boxplots of the full bootstrap distributions for all coefficients (similar as in Task 3).

Then, add (green) bars with the resulting standard errors to the bar plot produced in Task 3, allowing for comparison between Bootstrap and Bayesian standard errors. Interpret the results.

Answer:

Interpret the results

Nonparametric bootstrap usually does not use prior distribution; instead, it uses re-sampling from the empirical CDF. So it is quite good if we do not know about distribution, but if we know the value of the data set, it will be inaccurate as parametric bootstrap.

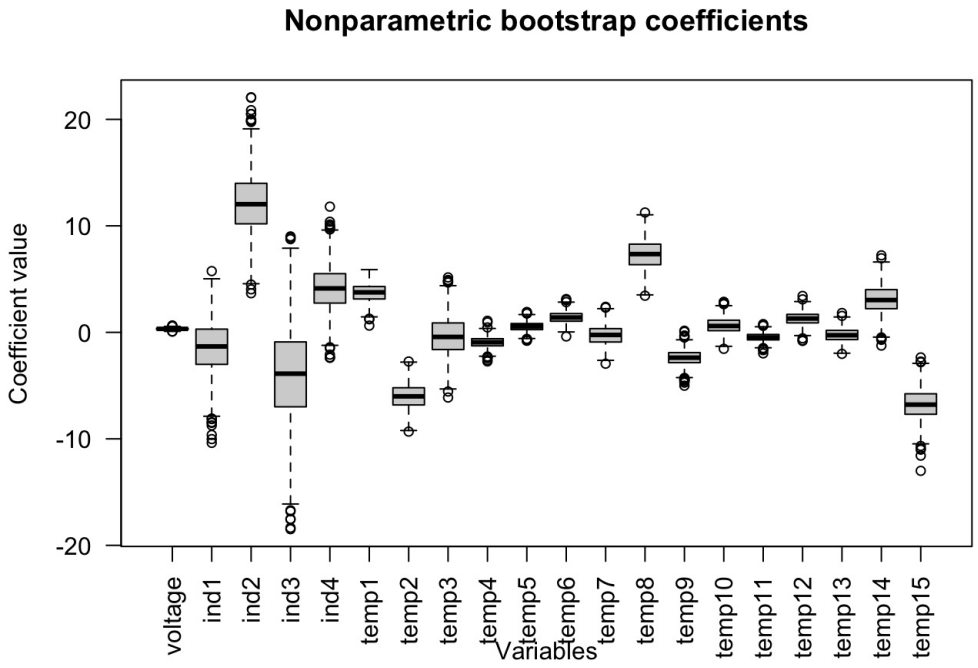
Nonparametric bootstrap estimating with linear regression has large standard deviation on coefficients than other methods. It is bigger than parametric linear regression. Usually, Nonparametric method is less powerful than the parametric method. Because they lose their relationships between variables, that means it is not easily interpreted through the predict function. So, according to the data, Nonparametric makes more errors in some coefficients.

I tried to compare estimating with linear regression and Lasso in Nonparametric method. If the Nonparametric bootstrap with lasso regression has a lower standard deviation than the linear regression model and Bayesian Lasso, it looks like a better way to choose a predicting model. However, the data of Nonparametric option has less power in interpreting the relationship in data. Therefore it is not proper to compare standard deviation with other methods. Especially Lasso estimating makes the variable selection, so the variable data will be reduced in each iteration which may make comparing standard deviation more unreliable.

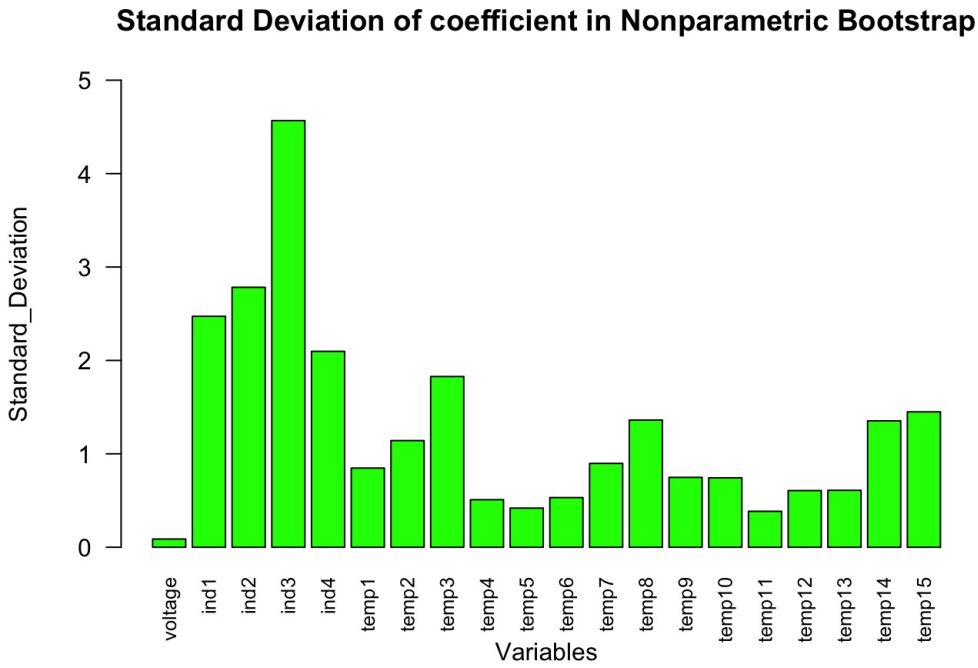
```
set.seed(1)
# Make bootstrap procedure in original data to make Nonparametric
# Set variables
B <- 1000
n <- dim(melter)[1]
Ynew <- matrix(0, n, B)
X <- model.matrix(vis.model)
Y <- Vis
Xnew<-list()
p<- dim(X)[2]
# Randomly choose the sample and regenerate data with Nonparametric
for (j in 1:B){
  Xnew[[j]]<- X
  for (i in 1:n){
    h<-sample(n,1)
    Xnew[[j]][i,]<- X[h,]
    Ynew[i,j] <-Y[h]
  }
}
# Set variable
all.real.boot <-matrix(0, B, p)
# Store the  linear regression model and coefficients of nonparametric data
for (i in 1:B){
  real.fitb<-lm(Ynew[,i]~Xnew[[i]][,2:21])
  all.real.boot[i,]<- summary(real.fitb)$coef[,1]
  # Fitting with Lasso for the test version
  #real.fitb<-glmnet(Xnew[[i]][,2:21], Ynew[,i], lambda=se.lambda, alpha = 1, family="gaussian")
  #temp<-t(as.matrix(coef(real.fitb)))
  #all.real.boot[i,]<- temp
}
# store data and calculate the standard deviation
all.boot2.sd<- apply(all.real.boot, 2, sd)
both.real.sd <- rbind( summary(vis.model)$coef[,2], all.boot2.sd)
both.real.sd
```

```
##              (Intercept)    voltage     ind1     ind2     ind3     ind4
##                718.5597 0.10974430 1.878539 2.358393 3.738723 2.366530
## all.boot2.sd    856.7013 0.08798095 2.472300 2.783040 4.567558 2.097099
##                   temp1    temp2    temp3     temp4     temp5     temp6
##              0.7669114 1.118409 1.732779 0.5024886 0.3662170 0.5361414
## all.boot2.sd 0.8476924 1.141577 1.827605 0.5089148 0.4192855 0.5315892
##                   temp7    temp8     temp9    temp10    temp11    temp12
##              0.7957883 1.345290 0.6152153 0.6645293 0.3882162 0.6538825
## all.boot2.sd 0.8974737 1.361655 0.7483017 0.7427784 0.3849289 0.6066899
##                  temp13   temp14   temp15
##              0.5949589 1.195092 1.254526
## all.boot2.sd 0.6094410 1.352831 1.449709
```

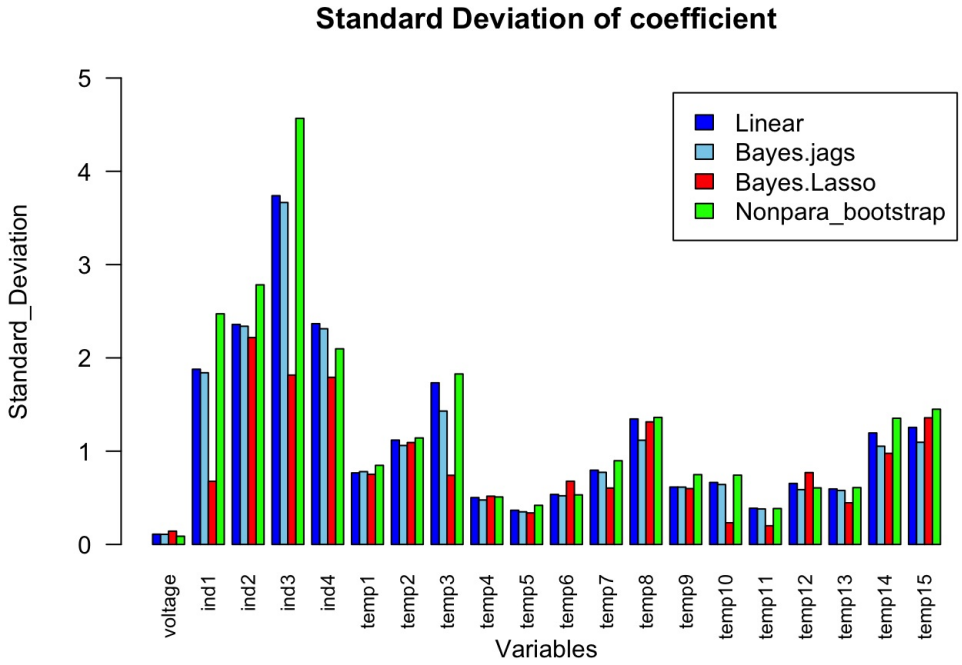
```
boot.coe<-all.real.boot[,2:21]
boot.sd<-all.boot2.sd[2:21]
# Boxplot of coefficients which is generated by nonparametric way
boxplot(boot.coe, names=xname,main='Nonparametric bootstrap coefficients', ylab="Coefficient value", xlab="Variab
les",las=2, cex = 0.8)
```




```
# Barplot of standard deviation of bootstrap way
barplot(boot.sd, names.arg=xname,ylim=c(0,5), col='green',main='Standard Deviation of coefficient in Nonparametri
c Bootstrap', ylab="Standard_Deviation", xlab="Variables",las=2, cex.names = 0.8)
```




```
# Make data frame for bar plot
df <- as.data.frame(cbind(melter.fit.sd, melter.bayes.sd, bay.sd, boot.sd))
val<- matrix(0, nrow = 4, ncol = 20)
# An empty matrix is a necessary requirement prior to copying data
val[1,] <- df$melter.fit.sd
val[2,] <- df$melter.bayes.sd
val[3,] <- df$bay.sd
val[4,] <- df$boot.sd
# Plot barplot to compare
barplot(val, names.arg = xname, beside = TRUE,ylim=c(0,5), col = c('blue','skyblue','red','green'), legend.text =
c("Linear",'Bayes.jags','Bayes.Lasso','Nonpara_bootstrap'),main='Standard Deviation of coefficient', ylab="Standa
rd_Deviation", xlab="Variables",las=2, cex.names = 0.8)
```




```
# Qestion and Conclustion
print('Interpret the results.')
```

```
## [1] "Interpret the results."
```

```
print('Nonparametric bootstrap usually does not use prior distribution; instead, it uses re-sampling from the emp
irical CDF. So it is quite good if we do not know about distribution, but if we know the value of the data set, i
t will be inaccurate as parametric bootstrap.')
```

```
## [1] "Nonparametric bootstrap usually does not use prior distribution; instead, it uses re-sampling from the em
pirical CDF. So it is quite good if we do not know about distribution, but if we know the value of the data set,
it will be inaccurate as parametric bootstrap."
```

```
print('Nonparametric bootstrap estimating with linear regression has large standard deviation on coefficients tha
n other methods. It is bigger than parametric linear regression. Usually, Nonparametric method is less powerful t
han the parametric method. Because they lose their relationships between variables, that means it is not easily i
nterpreted through the predict function. So, according to the data, Nonparametric makes more errors in some coeff
icients.')
```

```
## [1] "Nonparametric bootstrap estimating with linear regression has large standard deviation on coefficients th
an other methods. It is bigger than parametric linear regression. Usually, Nonparametric method is less powerful
than the parametric method. Because they lose their relationships between variables, that means it is not easily
interpreted through the predict function. So, according to the data, Nonparametric makes more errors in some coef
ficients."
```

```
print('I tried to compare estimating with linear regression and Lasso in Nonparametric method. If the Nonparametr
ic bootstrap with lasso regression has a lower standard deviation than the linear regression model and Bayesian L
asso, it looks like a better way to choose a predicting model. However, the data of Nonparametric option has less
power in interpreting the relationship in data. Therefore it is not proper to compare standard deviation with oth
er methods. Especially Lasso estimating makes the variable selection, so the variable data will be reduced in eac
h iteration which may make comparing standard deviation more unreliable.')
```

```
## [1] "I tried to compare estimating with linear regression and Lasso in Nonparametric method. If the Nonparamet
ric bootstrap with lasso regression has a lower standard deviation than the linear regression model and Bayesian
Lasso, it looks like a better way to choose a predicting model. However, the data of Nonparametric option has les
s power in interpreting the relationship in data. Therefore it is not proper to compare standard deviation with o
ther methods. Especially Lasso estimating makes the variable selection, so the variable data will be reduced in e
ach iteration which may make comparing standard deviation more unreliable."
```

Task 5: Model choice (10 marks)

Based on all considerations and analyses carried out so far, decide on a suitable model that you would present to a client, if you had been the statistical consultant.

Formulate the model equation in mathematical notation.

Refit your selected model using ordinary Least Squares. Carry out some residual diagnostics for this fitted model, and display the results. Discuss these briefly.

Answer:

1. Mathematical notation

Bayesian Lasso

For exact notation, according to the regression course slide R3, page 47 referenced the book 'Statistical Learning with Sparsity: The LASSO and Generalizations.' with section 6.1 expressing the Mathematical notation of Bayesian Lasso. It also referenced the approach of Park and Casella (2008), which paper wrote at the University of Florida, Gainesville, Florida, USA.

$y \mid \beta, \lambda, \sigma \sim N(X\beta, \sigma^2 I_{N\times N}) \cdots (1) \beta \mid \lambda, \sigma \sim p\prod j=1 \lambda 2\sigma e^{-\lambda\sigma|\beta_j|} \cdots (2)$ Laplacian prior (2). Under this model, it is easy to show that the negative log posterior density like under

$12\sigma^2 \|$ For \lambda fixed, with Laplacian prior density for \beta \begin{align} p(\beta)={1 \over (2r)^{p}}e^{(-{\lVert \beta\rVert_{1}\over r})} \end{align} under a normal response model \begin{align} y \sim N_{n}(X\beta,I_{n}\sigma^2) \end{align}

2. Carry out some residual diagnostics for this fitted model

From the residual plots it is not much different from the original Linear regression, but it looks more concentrated than the linear model because some of the variables were not selected, making residuals look more concentrated.

By the R squared point, refit Bayes Lasso model has small R squared than Linear regression. It means that the fitting is a little bit worse than full variable linear regression. But it covers 94% of the R squared point. That is, only 8 variables can cover many interpret of the dataset.

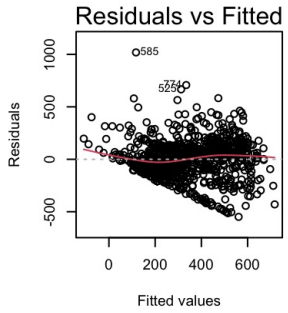
```
# Choose the model with Bayesian Lasso model  the reason because  Bayesian lasso have generally small standard de
veations in coefficients and with the MCMC process, it is easy to select some variables
# Select the variables with over 9000 counts it is about 90% of selection.
coe.val2<-coe2[which(coe2[,1] != 0)]
coe.ind2<-rownames(coe2)[which(coe2[,1] != 0)]
print(num[which(num$Count>=9000),1])
```

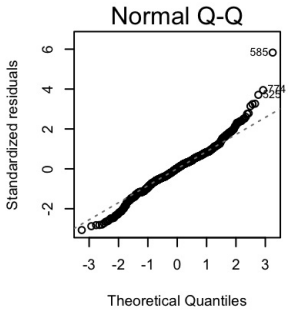
```
## [1] 9110 9996 9984 9997 9998 9991 9936 9983
```

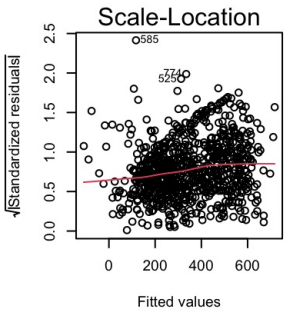
```
coe.val2<-num[which(num$Count>=9000),1]
coe.ind2<-rownames(num)[which(num$Count>=9000)]
print(coe.ind2)
```

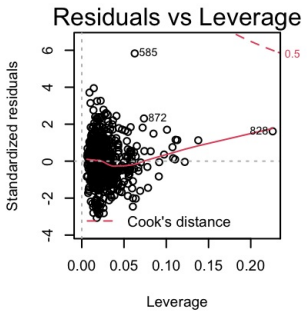
```
## [1] "voltage" "ind2"    "temp1"   "temp2"   "temp8"   "temp9"   "temp14"
## [8] "temp15"
```

```
# Do refit the selected variables with linear model and find the summary
refit.bayes.lasso<-lm(viscosity~voltage+ind2+temp1+temp2+temp8+temp9+temp14+temp15, data=melter)
summary.refit.bayes.lasso<-summary(refit.bayes.lasso)
# Residuals plots of linear regression and refit Bayesian Lasso and Q-Q plot for linear regression and Baysian La
sso model
op <- par(pty = "s", mfrow = c(2, 2),cex=0.6)
# Residual plot of linear regression
plot(vis.model)
```

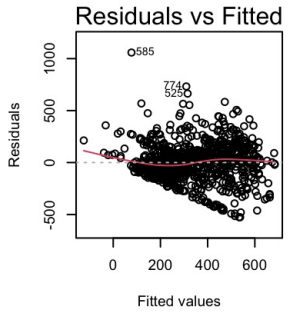


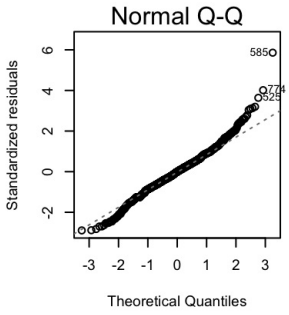


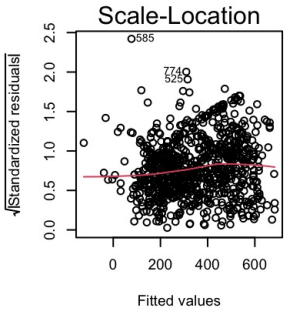


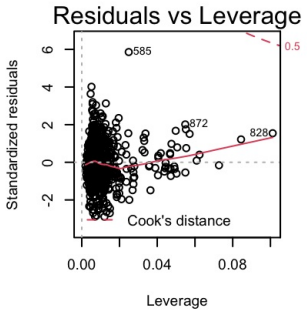



```
# Residual plot of selected variables fitted model
plot(refit.bayes.lasso)
```

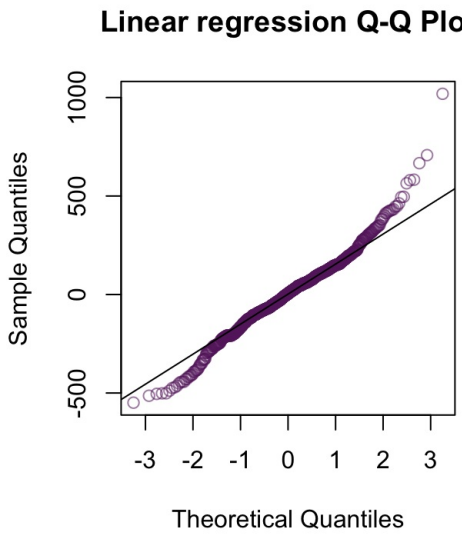


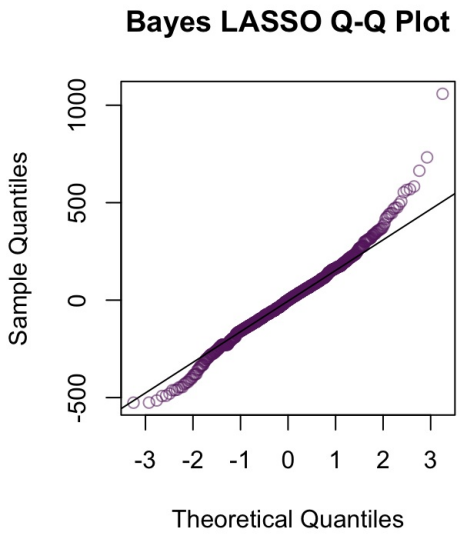







```
# QQ Plot of Linear regression and selected variables fitting
op <- par(pty = "s", mfrow = c(1, 2))
qqnorm(vis.model$residuals, col="#68246d90", main='Linear regression Q-Q Plot')
qqline(vis.model$residuals)
qqnorm(refit.bayes.lasso$residuals, col="#68246d90", main='Bayes LASSO Q-Q Plot')
qqline(refit.bayes.lasso$residuals)
```






```
# Find R squared score
print(vis.summary$r.squared)
```

```
## [1] 0.43212
```

```
print(vis.summary$adj.r.squared)
```

```
## [1] 0.4187581
```

```
print(summary.refit.bayes.lasso$r.squared)
```

```
## [1] 0.4082655
```

```
print(summary.refit.bayes.lasso$adj.r.squared)
```

```
## [1] 0.4027737
```

```
# Percentage of covering with selected variables
print(summary.refit.bayes.lasso$r.squared/vis.summary$r.squared*100)
```

```
## [1] 94.47966
```

```
# Question and conclusion
print('Discuss these briefly')
```

```
## [1] "Discuss these briefly"
```

```
print('From the residual plots it is not much different from the original Linear regression, but it looks more co
ncentrated than the linear model because some of the variables were not selected, making residuals look more conc
entrated.')
```

```
## [1] "From the residual plots it is not much different from the original Linear regression, but it looks more c
oncentrated than the linear model because some of the variables were not selected, making residuals look more con
centrated."
```

```
print('By the R squared score, refit Bayes Lasso model has small R squared than Linear regression. It means that
the fitting is a little bit worse than full variable linear regression. But it covers 94% of the R squared score.
That is, only 8 variables can cover many interpret of the dataset.')
```

```
## [1] "By the R squared score, refit Bayes Lasso model has small R squared than Linear regression. It means that
the fitting is a little bit worse than full variable linear regression. But it covers 94% of the R squared score.
That is, only 8 variables can cover many interpret of the dataset."
```

We will refer to the model produced in this task as (T5) henceforth.

Task 6: Extensions (20 marks)

For this task, take the model (T5) as the starting point. Then consider extensions of your model in TWO of the following THREE directions (of your choice).

1. Replace the temperature sensor variables in model (T5) by an adequate number of principal components (see Task 1).
2. Replace the `voltage`, and the remaining induction variables, by nonparametric terms.
3. Consider a transformation of the response variable `viscosity`.

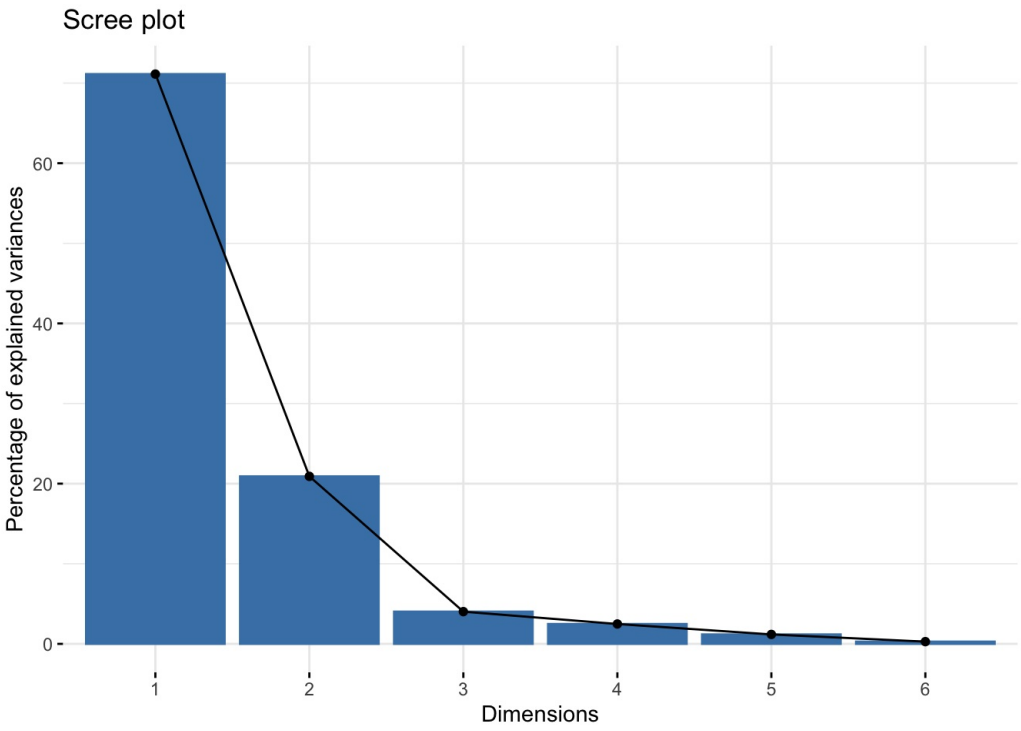
Each time, report the fitted model through adequate means. Discuss whether the corresponding extension is useful, giving quantitative or graphical evidence where possible.

Give a short discussion on whether any of your extensions have led to an actual improvement compared to model (T5).

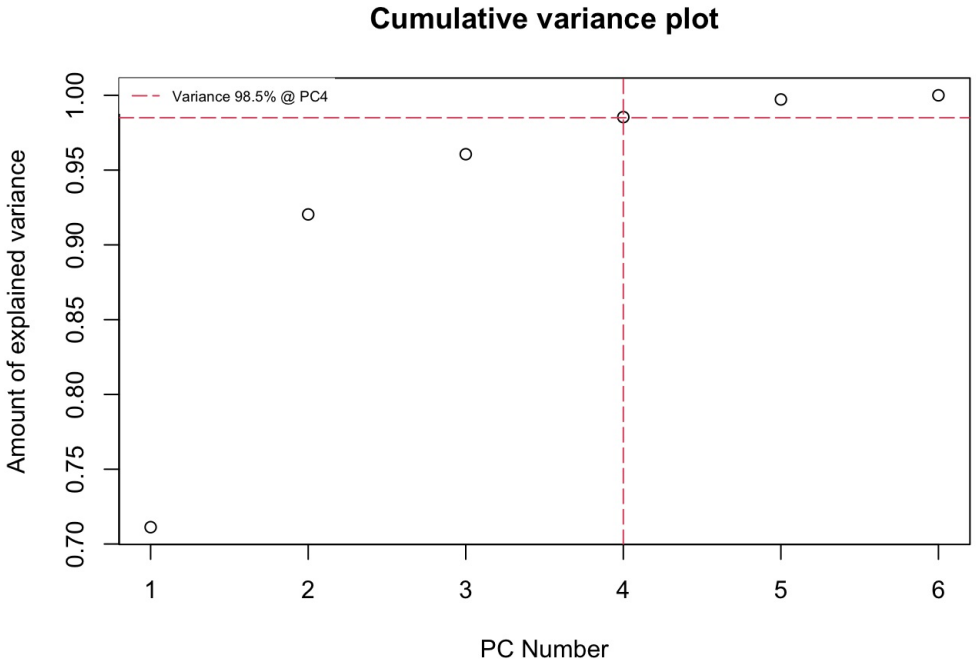
Answer:

I choose all three extensions for compare the results and write short report in each case in last part with some graphical plot.

```
# Replace the temperature sensor variables in model (T5) by an adequate number of principal components (see Task
1).
library("scatterplot3d")
# Reset the data with selected variables
re.melter2<-melter[c(coe.ind2)]
# Proceed the PCA with Temperature variables
pca.re.melter2<-prcomp(re.melter2[3:8], center = TRUE, scale = TRUE)
# Plot screeplot
fviz_eig(pca.re.melter2)
```



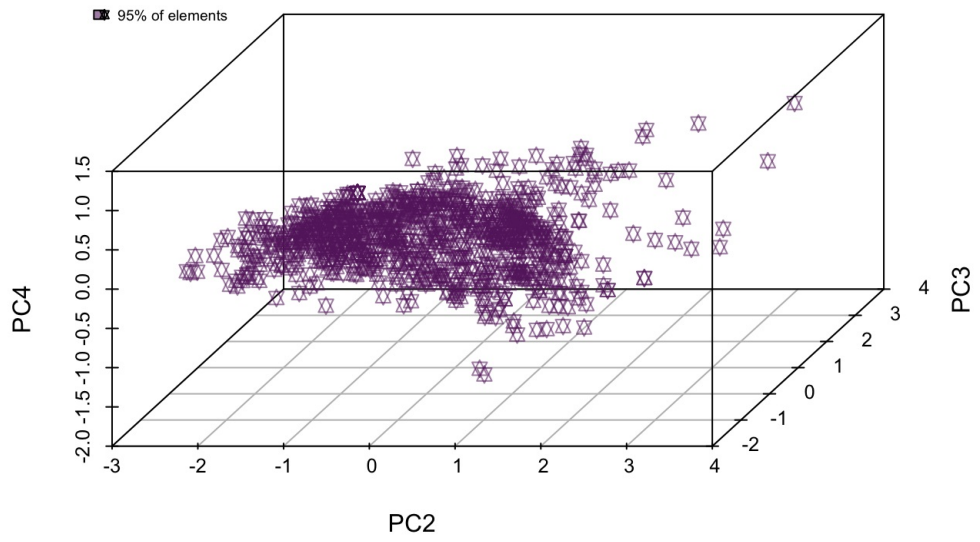

```
# Plot Cumulative variance
Lambda2 <- pca.re.melter2$sdev^2
cumpro2 <- cumsum(Lambda2 / sum(Lambda2))
plot(cumpro2, xlab = "PC Number", ylab = "Amount of explained variance", main = "Cumulative variance plot")
legend("topleft", legend=c("Variance 98.5% @ PC4"), col=c(2), lty=5, cex=0.6, box.lty=0)
abline(h = 0.985, col=2, lty=5)
abline(v = 4, col=2, lty=5)
```




```
# Get the variance of PCA and find contribution of variables
pca.var2 <- get_pca_var(pca.re.melter2)
contrib2 <- pca.var2$contrib
row.sum2<-rowSums(contrib2[,1:4])
# Add PCA elements in dataset
pc.melter2<-cbind(melter[,1],re.melter2,pca.re.melter2$x)
colnames(pc.melter2)[1]<-'viscosity'
# Get the summary of linear regression of PCA elements with viscosity
summary(lm(Vis~ PC1 + PC2 + PC3 + PC4, data=pc.melter2))
```

```
##
## Call:
## lm(formula = Vis ~ PC1 + PC2 + PC3 + PC4, data = pc.melter2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -601.40 -104.32   -3.00   91.21  981.02
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  343.000      6.567  52.234  < 2e-16 ***
## PC1           27.183      3.181   8.547  < 2e-16 ***
## PC2           93.189      5.866  15.887  < 2e-16 ***
## PC3          -98.554     13.371  -7.371 3.96e-13 ***
## PC4         -123.778     17.027  -7.269 8.07e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 193.8 on 866 degrees of freedom
## Multiple R-squared:  0.3331, Adjusted R-squared:  0.3301
## F-statistic: 108.2 on 4 and 866 DF,  p-value: < 2.2e-16
```

```
# Scatterplot
scatterplot3d(pc.melter2[,11:13], pch = 11, color='#68246d90' , angle = 60)
legend("topleft", legend=c('95% of elements'), fill=c('#68246d90'), pch=11, cex=0.6, box.lty=0)
```




```
# Discuss and Conclusion
print('Discuss whether the corresponding extension is useful, giving quantitative or graphical evidence where pos
sible.')
```

```
## [1] "Discuss whether the corresponding extension is useful, giving quantitative or graphical evidence where po
ssible."
```

```
print('The benefit of PCA compared to Lasso is that PCA does not require fitting a high-dimensional model, and PC
A can be applied to the multi-dimensional space before any regression takes place. If you see the 3D scatter plot
, it already contains almost 95% value of multi-variables just in 3D space.')
```

```
## [1] "The benefit of PCA compared to Lasso is that PCA does not require fitting a high-dimensional model, and P
CA can be applied to the multi-dimensional space before any regression takes place. If you see the 3D scatter plo
t, it already contains almost 95% value of multi-variables just in 3D space."
```

```
print('Lasso has the advantage that its variable selection is more interpretable, and the selected variables corr
espond to the original variables. So unlike PCA, selected variables are not easy to show graphically.')
```

```
## [1] "Lasso has the advantage that its variable selection is more interpretable, and the selected variables cor
respond to the original variables. So unlike PCA, selected variables are not easy to show graphically."
```

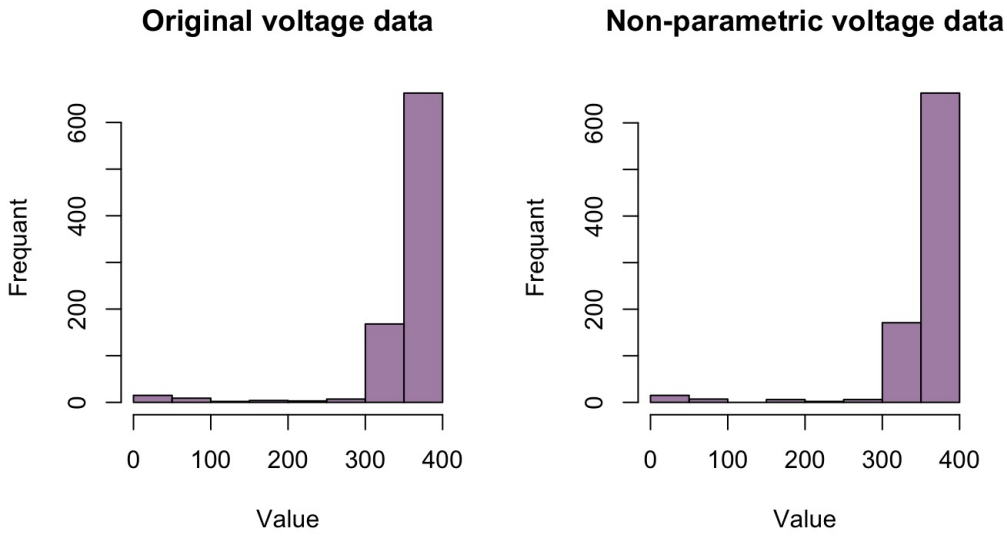
```
print('The PCA  prediction has less powerful than Bayesian Lasso selected variable prediction. It means PCA can c
over multi-dimensional variables easily, but it does not easily interpret the relationship between variables.')
```

```
## [1] "The PCA  prediction has less powerful than Bayesian Lasso selected variable prediction. It means PCA can
cover multi-dimensional variables easily, but it does not easily interpret the relationship between variables."
```

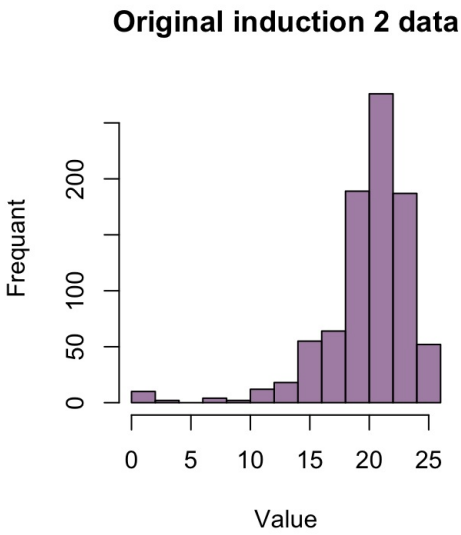
```
# Replace the voltage, and the remaining induction variables, by nonparametric terms.
# Nonparametric dataset build
B <- 1
n <- dim(re.melter2)[1]
X <- model.matrix(lm(Vis~.,data=re.melter2))
Xnew2<-list()
p<- dim(X)[2]
for (j in 1:B){
  Xnew2[[j]]<- X
  for (i in 1:n){
    h<-sample(n,1)
    Xnew2[[j]][i,]<- X[h,]
  }
}
# Cbind with Nonparametric Voltage and Induction2
new.re.melter2<-as.data.frame(cbind(Xnew2[[1]][,2:3],re.melter2[,3:8]))
# Get the summary of linear regression of Nonparametric way
summary(lm(Vis~.,data=new.re.melter2))
```

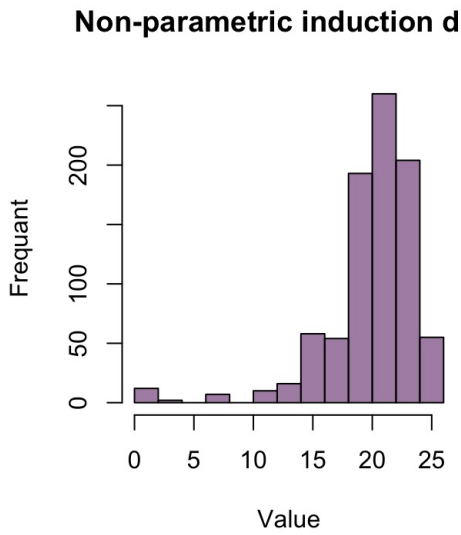
```
##
## Call:
## lm(formula = Vis ~ ., data = new.re.melter2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -550.70 -115.70    3.03  103.13  949.67
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.53127  394.97237   0.080 0.936390
## voltage      -0.07185    0.11487  -0.625 0.531832
## ind2          1.54318    1.68797   0.914 0.360857
## temp1         2.30075    0.60818   3.783 0.000166 ***
## temp2        -4.80473    0.74950  -6.411 2.38e-10 ***
## temp8         6.03510    0.98624   6.119 1.42e-09 ***
## temp9        -3.43988    0.49389  -6.965 6.52e-12 ***
## temp14        5.24467    0.53346   9.832  < 2e-16 ***
## temp15       -5.16027    1.12041  -4.606 4.73e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 187.6 on 862 degrees of freedom
## Multiple R-squared:  0.3781, Adjusted R-squared:  0.3724
## F-statistic: 65.52 on 8 and 862 DF,  p-value: < 2.2e-16
```

```
# Plot histogram of parametric and nonparametric for compare
op <- par(pty = "s", mfrow = c(1, 2))
hist(melter$voltage, main='Original voltage data', xlab='Value',ylab='Frequant', col='#68246d90')
hist(new.re.melter2$voltage, main='Non-parametric voltage data', xlab='Value',ylab='Frequant', col='#68246d90')
```




```
hist(melter$ind2, main='Original induction 2 data', xlab='Value',ylab='Frequant', col='#68246d90')
hist(new.re.melter2$ind2, main='Non-parametric induction data', xlab='Value',ylab='Frequant', col='#68246d90')
```





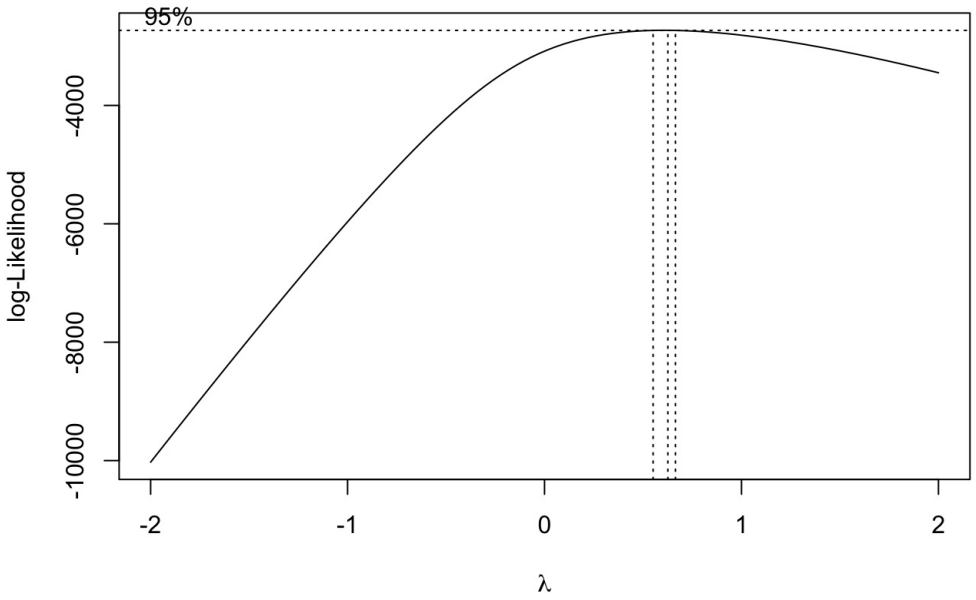

```
# Discuss and Conclusion
print('Discuss whether the corresponding extension is useful, giving quantitative or graphical evidence where pos
sible.')
```

```
## [1] "Discuss whether the corresponding extension is useful, giving quantitative or graphical evidence where po
ssible."
```

```
print('Nonparametric in voltage and induction also cannot interpret the relationship with other variables, but if
it uses a histogram plot, it could represent almost the same distribution as a parametric one because it is not d
ependent on different parameters. In the R square score, the refit Bayesian Lasso has a better score. It is the s
imilar result with task4 which is related to parametric and nonparametric. It cannot interpret the relationship b
etween other variables, so the R square is smaller than refit Lasso which means a poor fit than refit lasso.')
```

```
## [1] "Nonparametric in voltage and induction also cannot interpret the relationship with other variables, but i
f it uses a histogram plot, it could represent almost the same distribution as a parametric one because it is not
dependent on different parameters. In the R square score, the refit Bayesian Lasso has a better score. It is the
similar result with task4 which is related to parametric and nonparametric. It cannot interpret the relationship
between other variables, so the R square is smaller than refit Lasso which means a poor fit than refit lasso."
```

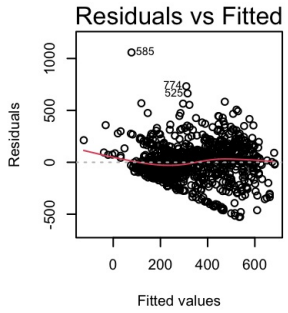
```
# Consider a transformation of the response variable viscosity.
library(MASS)
# Find optimal lambda for Box-Cox transformation
co2<-lm(Vis~.,data=new.re.melter2)
bc2 <- boxcox(Vis~.,data=re.melter2)
```

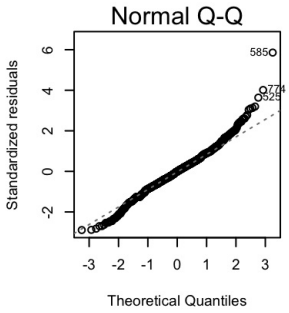


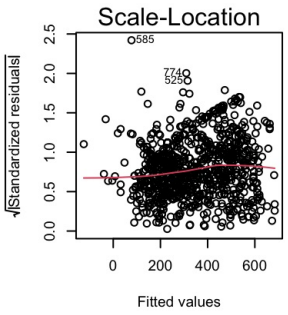

```
bc.lambda2 <- bc2$x[which.max(bc2$y)]
# Fit new linear regression model using the Box-Cox transformation with lambda !=0
bc.time2<-as.data.frame(cbind(((Vis^bc.lambda2-1)/bc.lambda2),as.matrix(re.melter2)))
bc.model2 <- lm(V1~.,data=bc.time2)
# Get summary of linear regression of BoxCox transformation
summary(bc.model2)
```

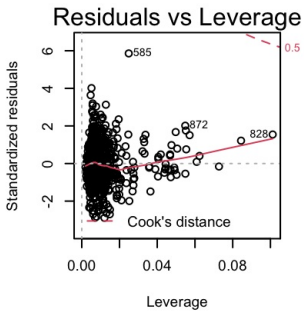
```
##
## Call:
## lm(formula = V1 ~ ., data = bc.time2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -74.427 -13.079   1.799  13.967 104.309
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -46.19911   47.49028  -0.973  0.33092
## voltage       0.03731    0.01336   2.793  0.00533 **
## ind2          1.31417    0.25166   5.222 2.22e-07 ***
## temp1         0.51386    0.07636   6.729 3.11e-11 ***
## temp2        -0.85738    0.09693  -8.846  < 2e-16 ***
## temp8         0.78347    0.11881   6.595 7.43e-11 ***
## temp9        -0.33210    0.06284  -5.285 1.59e-07 ***
## temp14        0.61274    0.06433   9.525  < 2e-16 ***
## temp15       -0.66123    0.13405  -4.933 9.72e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.47 on 862 degrees of freedom
## Multiple R-squared:  0.3913, Adjusted R-squared:  0.3856
## F-statistic: 69.25 on 8 and 862 DF,  p-value: < 2.2e-16
```

```
# Plot residuals
op <- par(pty = "s", mfrow = c(2, 2),cex=0.6)
plot(refit.bayes.lasso)
```

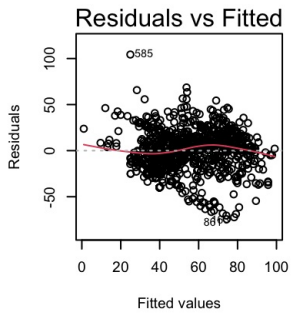


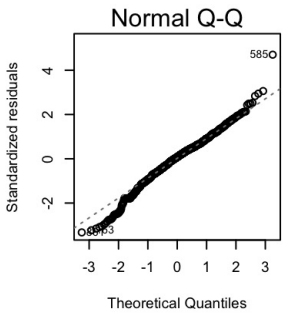


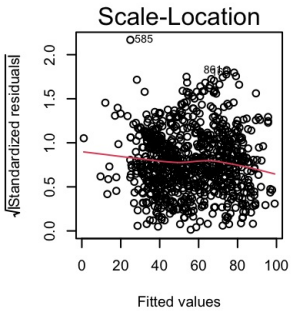


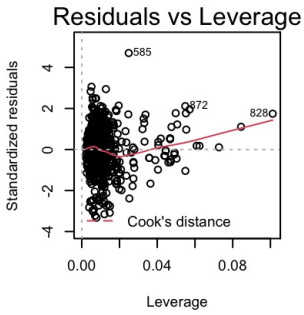



```
plot(bc.model2)
```










```
print(summary(bc.model2)$r.squared)
```

```
## [1] 0.3912555
```

```
print(summary(bc.model2)$adj.r.squared)
```

```
## [1] 0.3856059
```

```
# Discuss and Conclusion
print('From the refit Bayesian Lasso residual plot, it looks like Heteroskedasticity in the Residual vs Fitted pl
ot, which means the variance of the residuals is unequal thru measured values. Like viscosity data in this datase
t, a wide range of values is prone to heteroskedasticity. If heteroskedasticity exists, the regression contains u
nequal variance, which leads to the analysis of fitting would be poor.')
```

```
## [1] "From the refit Bayesian Lasso residual plot, it looks like Heteroskedasticity in the Residual vs Fitted p
lot, which means the variance of the residuals is unequal thru measured values. Like viscosity data in this datas
et, a wide range of values is prone to heteroskedasticity. If heteroskedasticity exists, the regression contains
unequal variance, which leads to the analysis of fitting would be poor."
```

```
print('In the BoxCox transformation, the response variable will be transformed to a more general form that occurs
with a transform parameter called a lambda value. It helps stabilise variance, make the data more normal distribu
tion-like, and improve the validity of data stabilisation procedures.
Usually, BoxCox can run on over 0 values (In the R function). In this model, the viscosity should be over 0 by th
e second law of thermodynamics. So It can use BoxCox transformation (Therefore, it is crucial to refine the datas
et before interpreting)')
```

```
## [1] "In the BoxCox transformation, the response variable will be transformed to a more general form that occur
s with a transform parameter called a lambda value. It helps stabilise variance, make the data more normal distri
bution-like, and improve the validity of data stabilisation procedures.\nUsually, BoxCox can run on over 0 values
(In the R function). In this model, the viscosity should be over 0 by the second law of thermodynamics. So It can
use BoxCox transformation (Therefore, it is crucial to refine the dataset before interpreting)"
```

```
print('In this transformation case, it makes residual plots more evenly (more normal distributed). This could fin
d the fact that the residual plots between refit Bayesian Lasso and BoxCox transformation.
Box-Cox is a little bit successful in reducing heteroskedasticity, but the inference of model fitting was unsucce
ssful. Moreover, the Box-Cox transformation interpreted the coefficients of the reduced model to complicate by ch
anging the data. So R square score was slightly lower than the original reduced fit data.')
```

```
## [1] "In this transformation case, it makes residual plots more evenly (more normal distributed). This could fi
nd the fact that the residual plots between refit Bayesian Lasso and BoxCox transformation.\nBox-Cox is a little
bit successful in reducing heteroskedasticity, but the inference of model fitting was unsuccessful. Moreover, the
Box-Cox transformation interpreted the coefficients of the reduced model to complicate by changing the data. So R
square score was slightly lower than the original reduced fit data."
```

```
print('Overall, three methods, all of which make the R square score, were reduced. It means that our model was fi
tted worse. Furthermore, I can understand that there are many fast and easy ways to infer the data but using orig
inal data to fitting is the most accurate.')
```

```
## [1] "Overall, three methods, all of which make the R square score, were reduced. It means that our model was f
itted worse. Furthermore, I can understand that there are many fast and easy ways to infer the data but using ori
ginal data to fitting is the most accurate."
```