**Human freedom predicting Model**

1. Introduction

Human freedom has 3 independent variables(hf\_score, hf\_rank, hf\_quartile) that are dependent on the other 115 independent variables. At the first glance, we can’t know that which dependent variables are most significant predictor for them.

1. Data analysis and minimizing

It has so many independent variables that we should consider that there are correlations between them.

- data loading and cleaning the NA values

data = read.csv('human-freedom-index-data.csv')

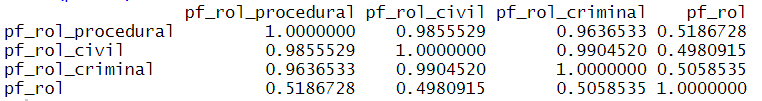
data[is.na(data)] = 0

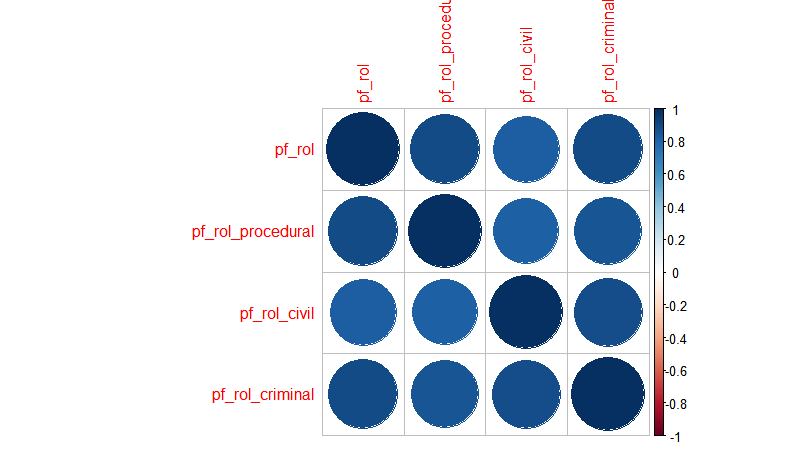
To calculate the correlation, we should clean the NA values. Omitting the NA values is simple but not good way because it could cause the loss of information. Therefore, we insert 0 value into the NA. As you can see, the data consists of only numbers so it is good solution.

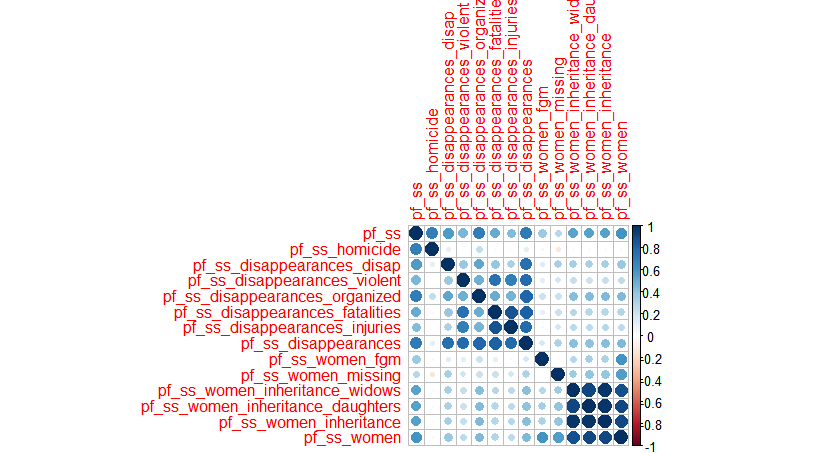
- correlation

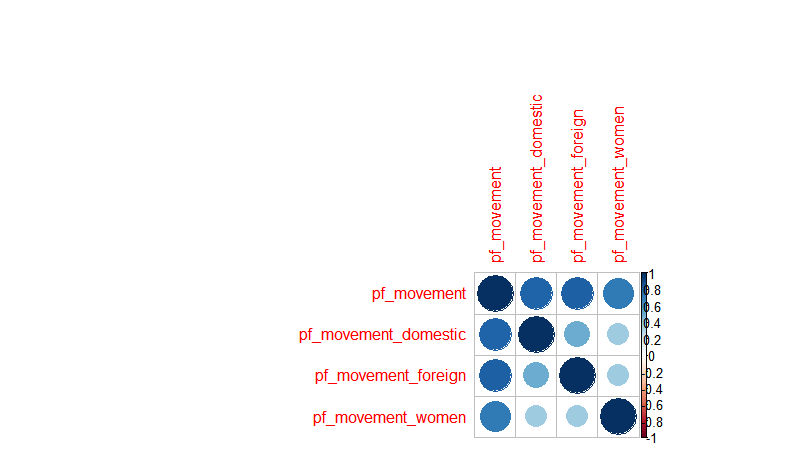
Calculating all correlations over all independent variables is ideal and theoretical way, but it takes some time. In this case, its calculation is 2^115. That is too boring and not optimized way. Here, we get the several partial correlation and then finally total correlation.

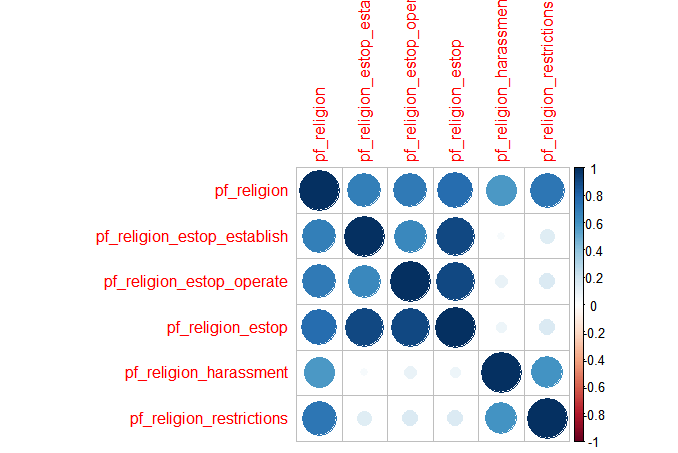
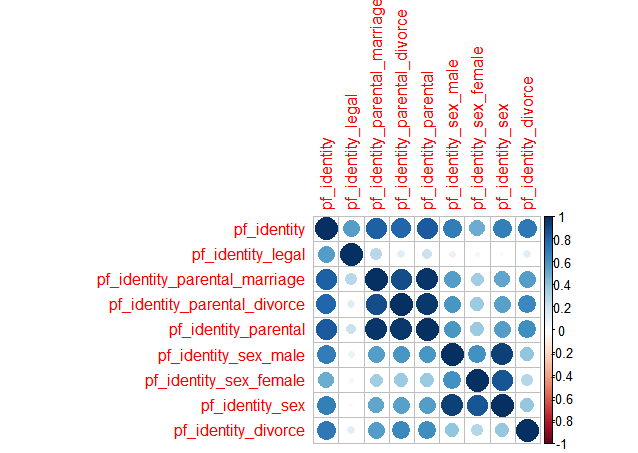
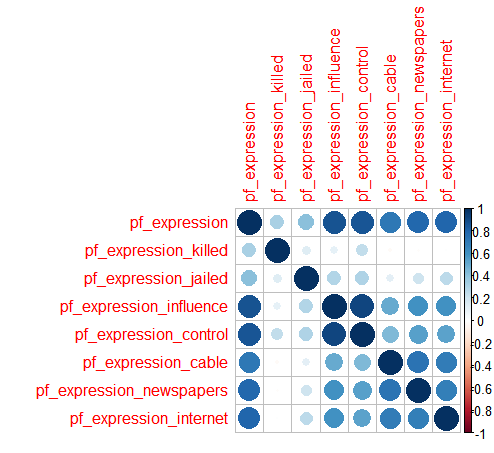
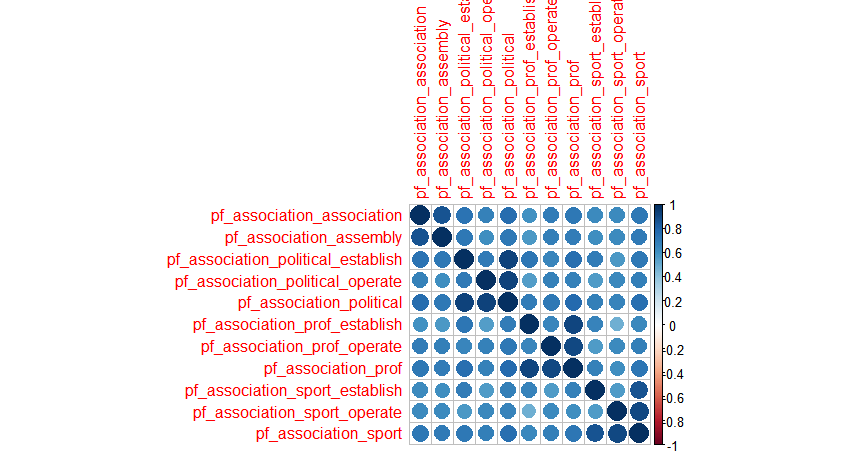
As you can see, independent variables consist of several categories such as pf\_rol, pf\_ss, pf\_movement, ef\_legl and etc. We can get the partial correlation of them.

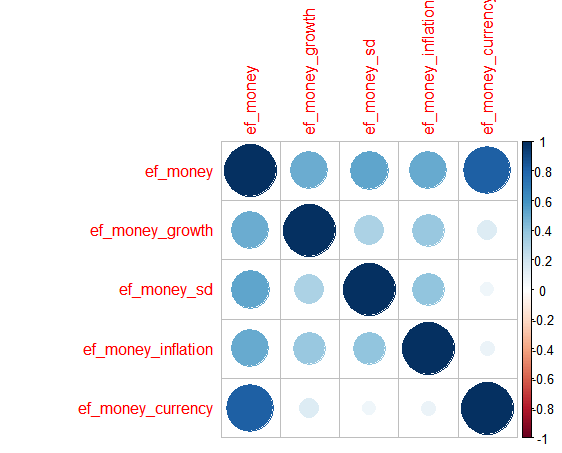


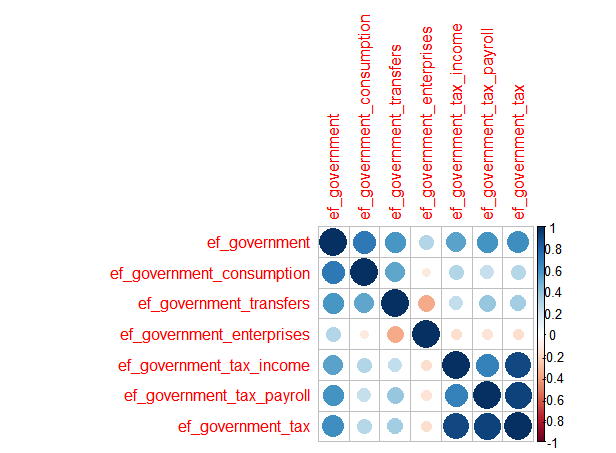


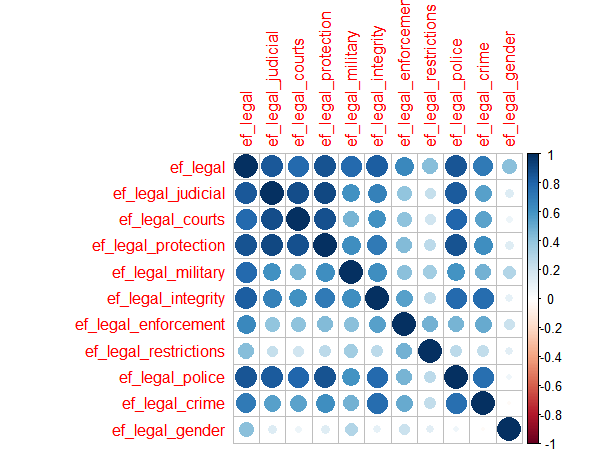


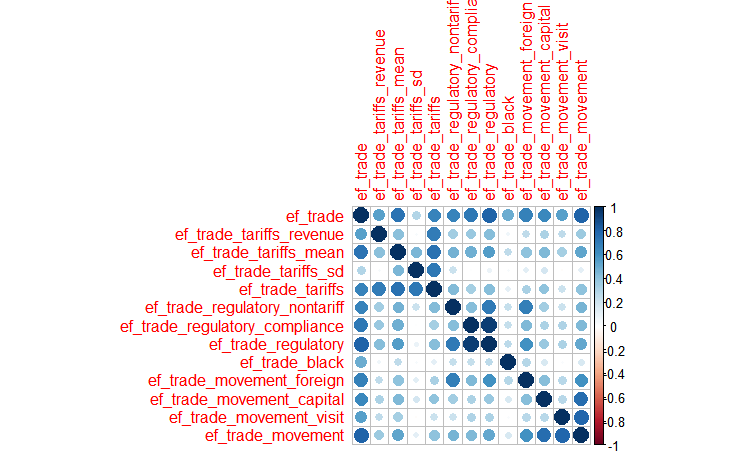


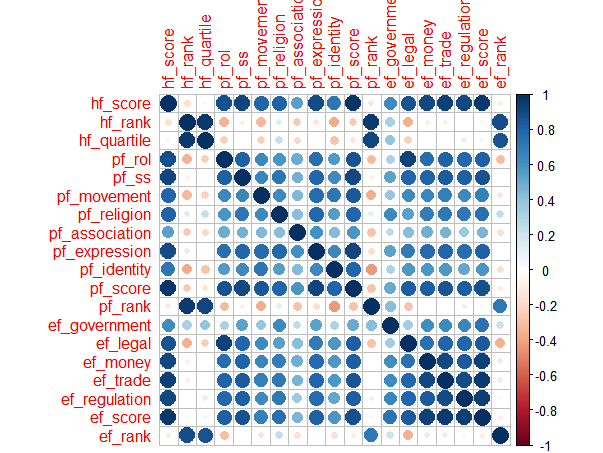












(The bluer, the more correlation)

We can concatenate several independent variables having the high correlation values (here higher than 0.95).

- minimize the independent variables

As I mentioned before, we should find all independent variables that have correlation value is higher than 0.95.

Following the independent variables are highly correlated and have similar type of values so I minimize them like this way.

data['pf\_ss\_women\_inheritance\_widows'] = (data['pf\_ss\_women\_inheritance\_widows'] + data['pf\_ss\_women\_inheritance\_daughters']) / 2

data['pf\_rol\_civil'] = (data['pf\_rol\_procedural'] + data['pf\_rol\_civil'] + data['pf\_rol\_criminal']) / 3

data['pf\_ss\_disappearances'] = (data['pf\_ss\_disappearances'] + data['pf\_ss\_disappearances\_fatalities'] + data['pf\_ss\_disappearances\_injuries'] ) / 3

data['pf\_religion\_estop\_operate'] = (data['pf\_religion\_estop\_operate'] + data['pf\_religion\_estop\_establish']) / 2

data['pf\_religion'] = (data['pf\_religion'] + data['pf\_religion\_harassment']) / 2

data['pf\_association\_association'] = (data['pf\_association\_association'] + data['pf\_association\_assembly']) / 2

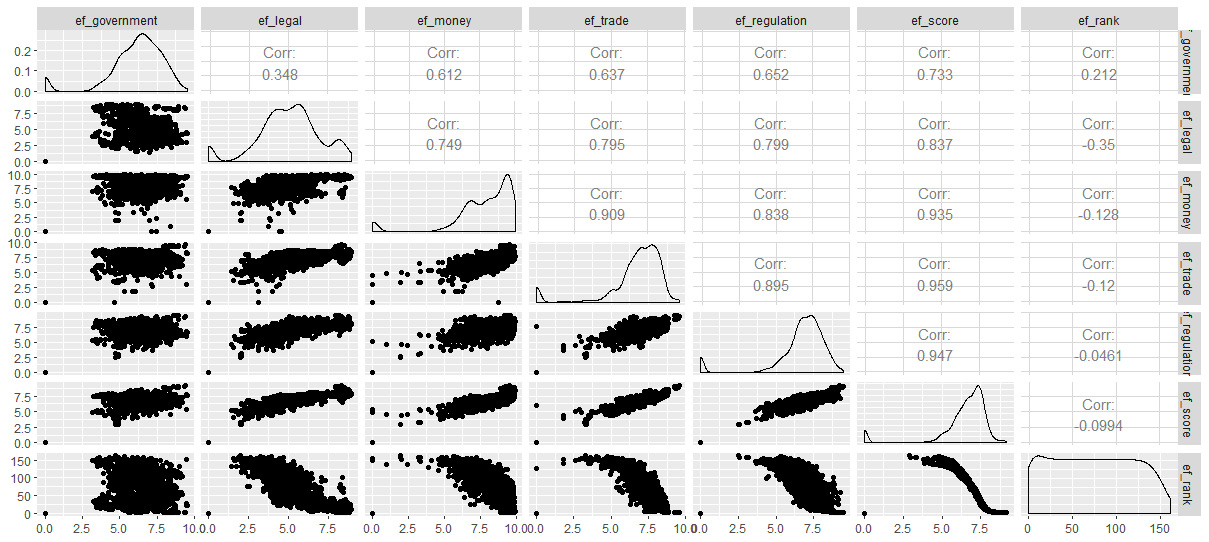
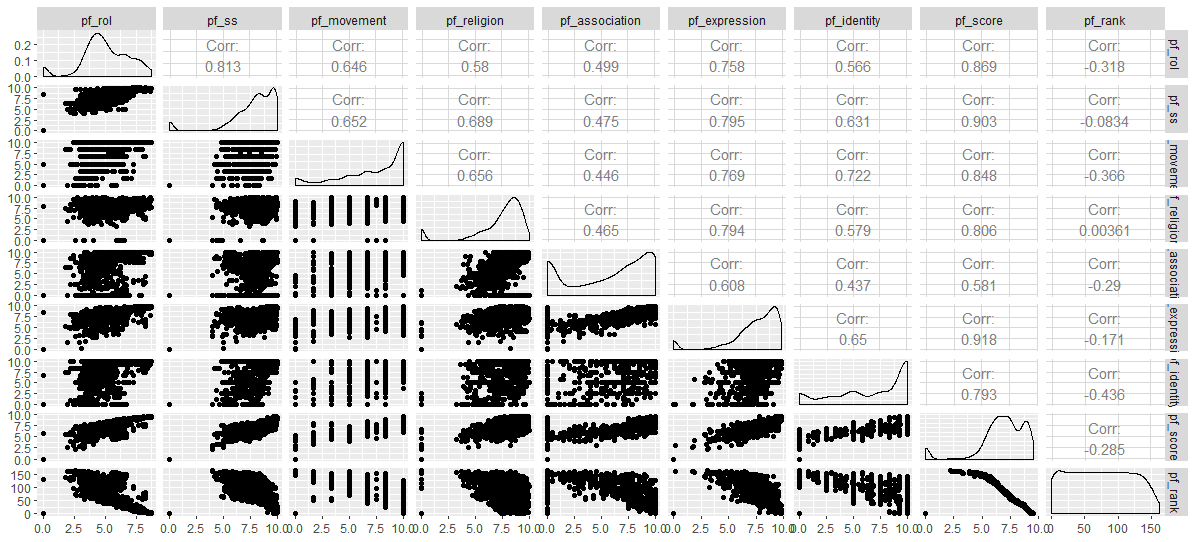
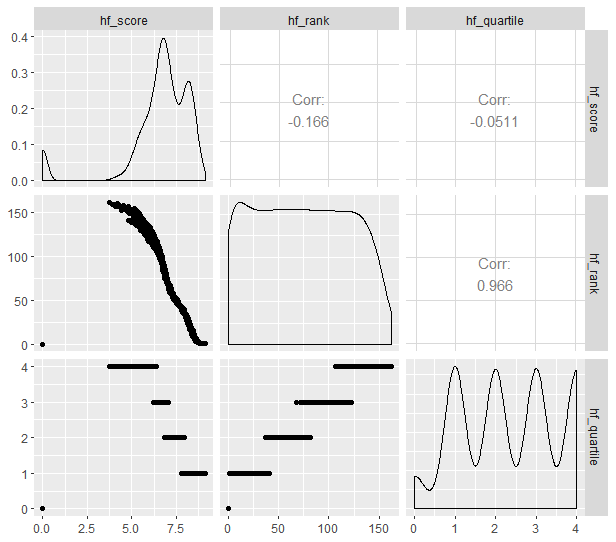
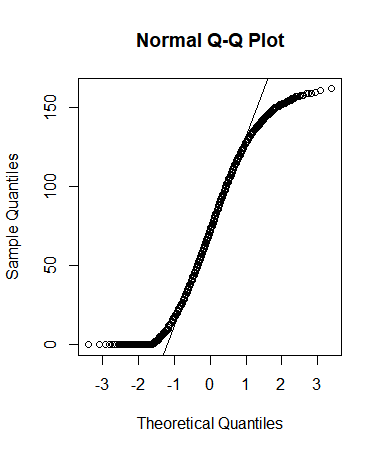
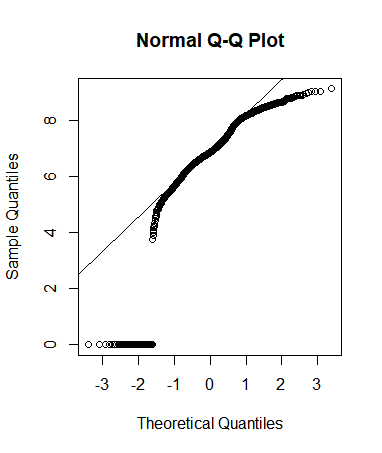
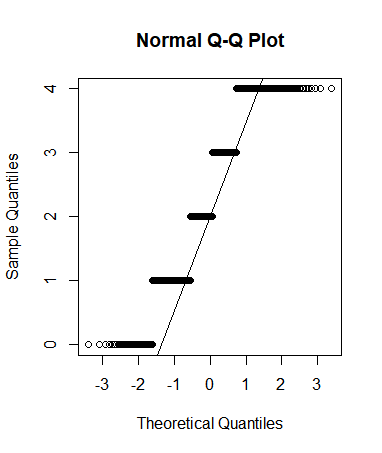
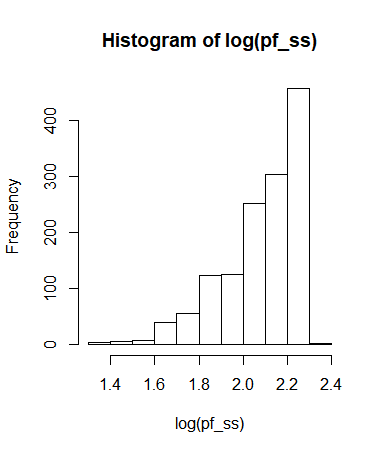
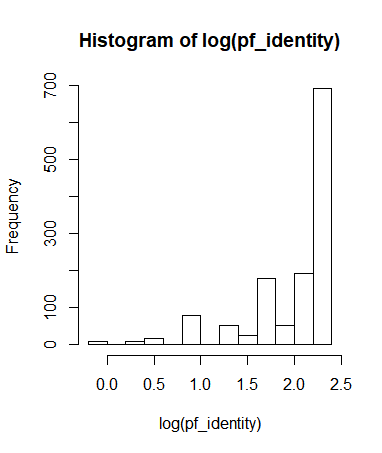
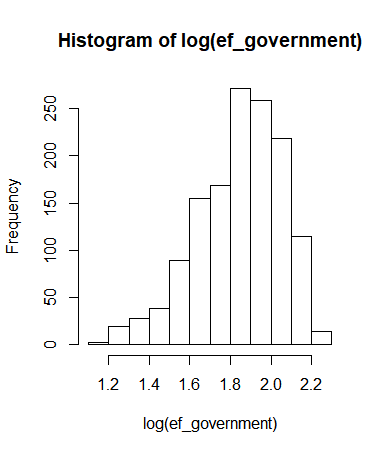
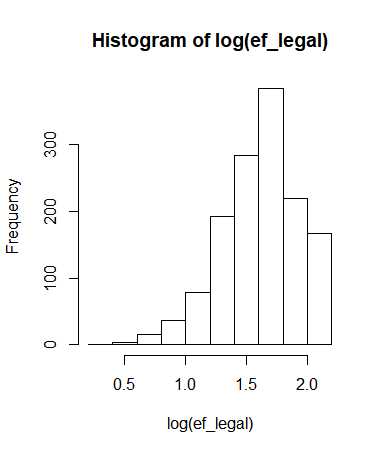
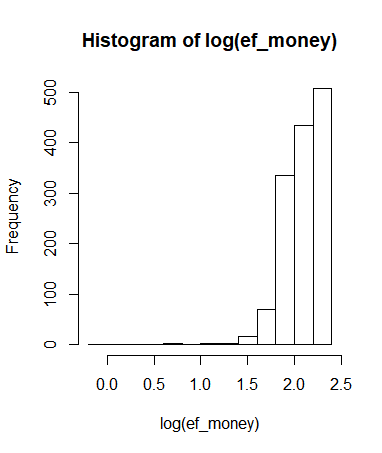
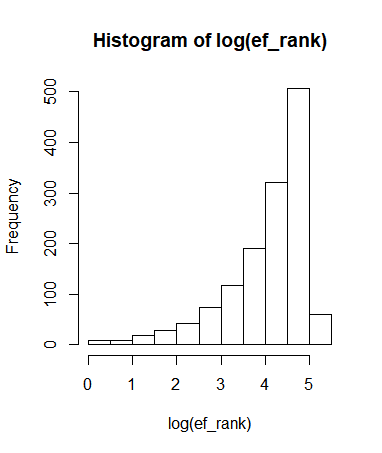
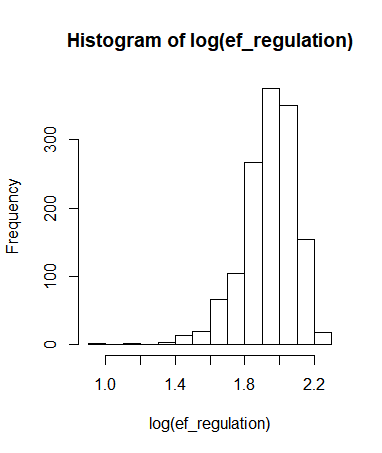
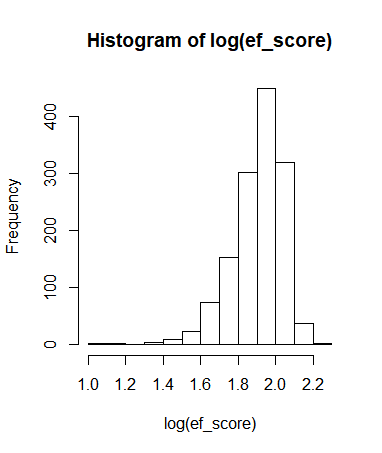
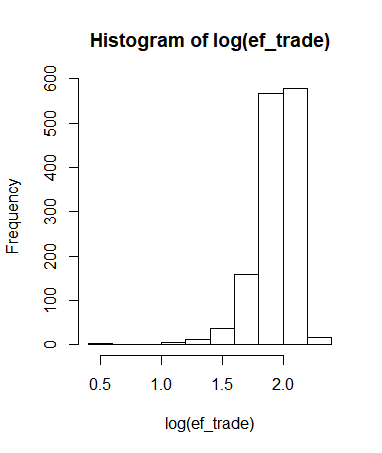
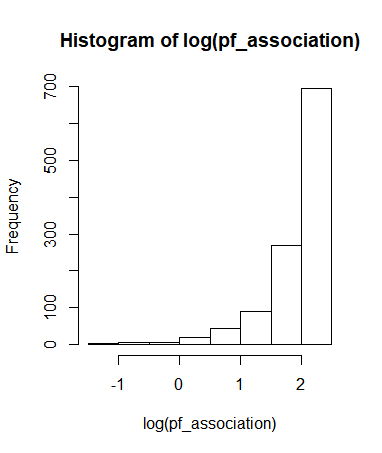
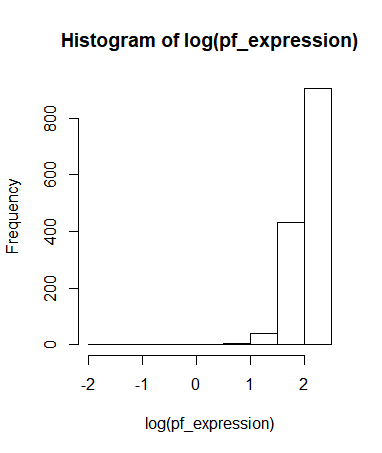
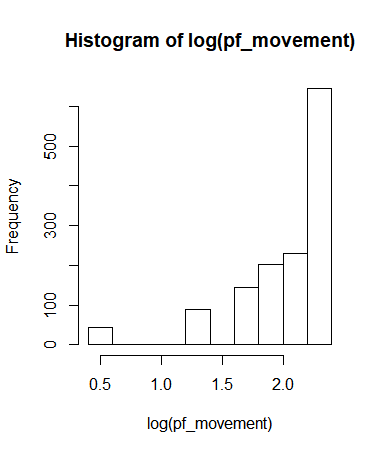
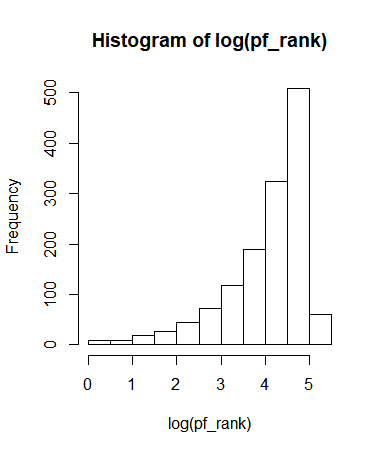
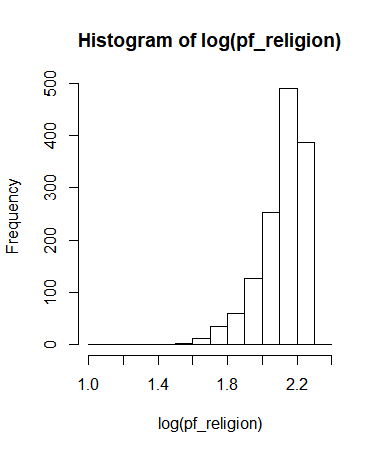
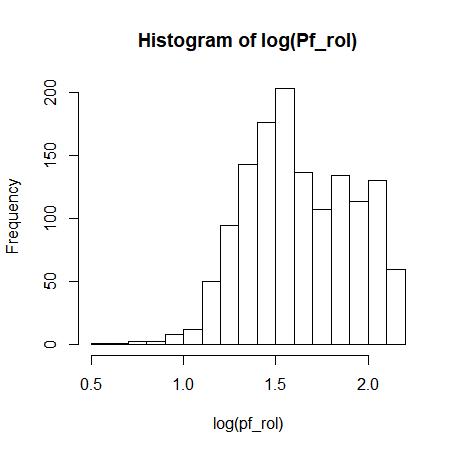
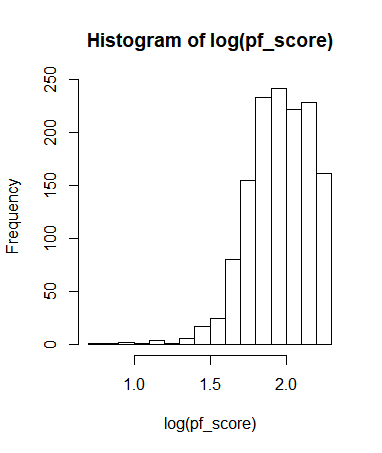
data['pf\_expression\_newspapers'] = (data['pf\_expression\_newspapers'] + data['pf\_expression\_cable']) / 2

data['pf\_identity\_parental\_marriage'] = (data['pf\_identity\_parental\_marriage'] + data['pf\_identity\_parental\_divorce']) / 2

data['pf\_identity\_sex'] = (data['pf\_identity\_sex'] + data['pf\_identity\_sex\_male'] + data['pf\_identity\_sex\_female']) / 3

data['ef\_government\_tax'] = (data['ef\_government\_tax'] + data['ef\_government\_tax\_income']) / 2

data['ef\_trade\_regulatory'] = (data['ef\_trade\_regulatory'] + data['ef\_trade\_regulatory\_compliance']) / 2



1. Data partition

Data partitioning is common in statistically getting the predicting model. It means that we divide data into training part and testing part.

train : test = 7:3 or 8:2

ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))

train <- data[ind==1,]

train <- train[c(names(train)[6:124])]

test <- data[ind==2,]

test <- test[c(names(test)[6:124])]

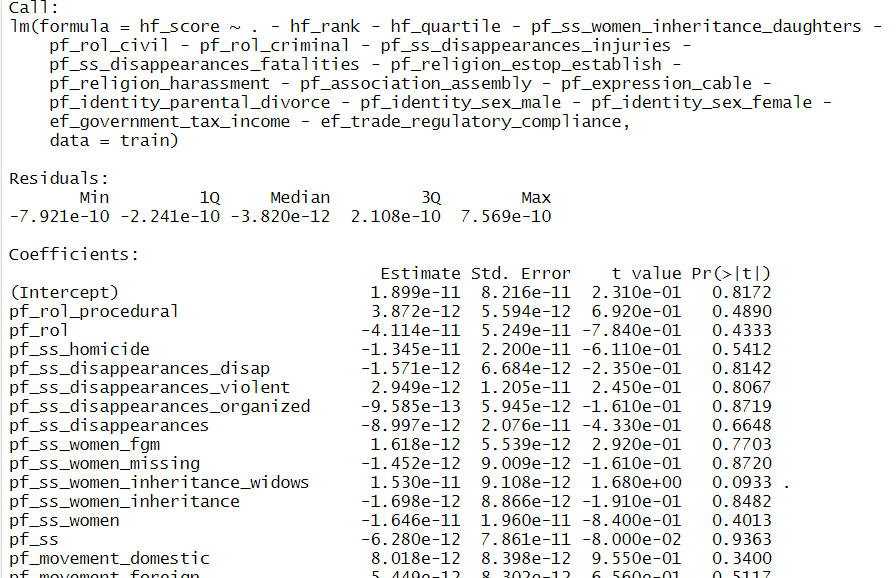
1. Predicting models

From the data, we can know that hf\_score and hf\_rank are continuous variables. Therefore, linear regression is the solution for the hf\_score and hf\_rank.

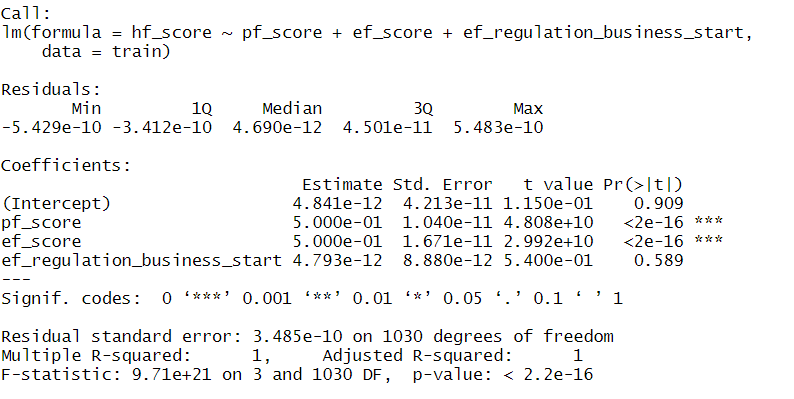
Linear regression is used to predict the value of an outcome variable Y based on one or more input predictor variables X. The aim is to establish a linear relationship (a mathematical formula) between the predictor variable(s) and the response variable, so that, we can use this formula to estimate the value of the response Y, when only the predictors (Xs) values are known.

hf\_score\_fullModel = lm(hf\_score ~ .- hf\_rank - hf\_quartile - pf\_ss\_women\_inheritance\_daughters-pf\_rol\_civil-pf\_rol\_criminal-pf\_ss\_disappearances\_injuries-pf\_ss\_disappearances\_fatalities-pf\_religion\_estop\_establish-pf\_religion\_harassment-pf\_association\_assembly-pf\_expression\_cable-pf\_identity\_parental\_divorce-pf\_identity\_sex\_male-pf\_identity\_sex\_female-ef\_government\_tax\_income - ef\_trade\_regulatory\_compliance ,train )

summary(hf\_score\_fullModel)



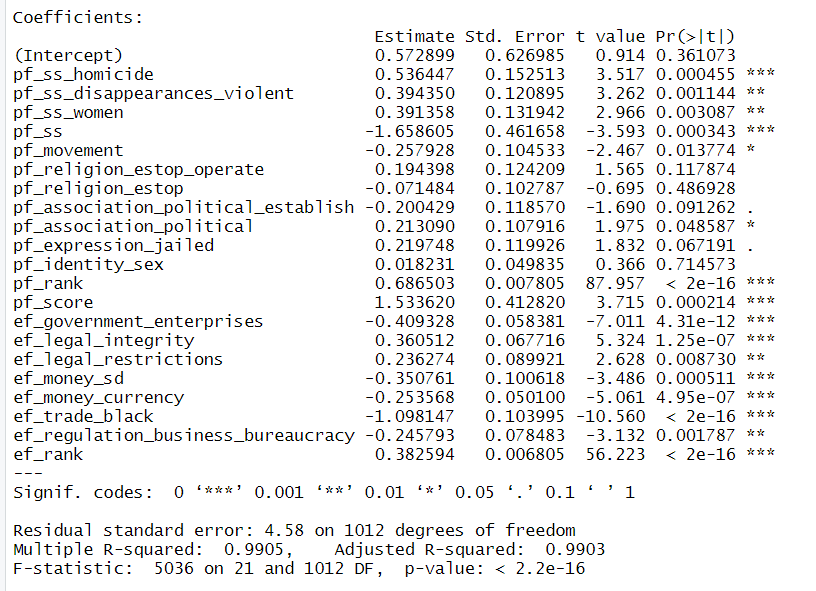
Here we choose only some variables with p-values less than 0.05 that are highly significant to the model.



**hf\_score\_reducedModel = lm(hf\_score ~ pf\_score + ef\_score, train)**

In this way, we got the hf\_rank model.

**hf\_rank\_reducedModel = lm(hf\_rank ~ pf\_ss\_homicide + pf\_ss\_disappearances\_violent + pf\_ss\_women + pf\_ss + pf\_movement + pf\_rank + pf\_score + ef\_government\_enterprises + ef\_legal\_integrity + ef\_legal\_restrictions + ef\_money\_sd + ef\_money\_currency + ef\_trade\_black + ef\_regulation\_business\_bureaucracy + ef\_rank, train)**



Let’s talk about hf\_quartile model that is special in this project.

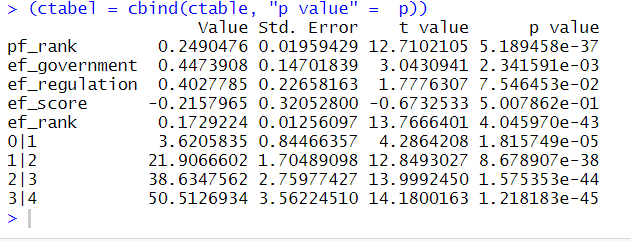
Its dependent variable is discrete value and has only 5 values(0(NA), 1, 2, 3, 4).

So the ordinary logistic regression model is suitable. This is a statistical technique that is used to predict behavior of ordinal level dependent variables with a set of independent variables.  The dependent variable is the order response category variable and the independent variable may be categorical or continuous.

Here you can see ‘fixed-polr.R’ that is also used for ordinal logistic regression.

hf\_quartilerducedModel <- polr(hf\_quartile ~ pf\_rank + ef\_government + ef\_regulation + ef\_score + ef\_rank, data = train, Hess = TRUE )

summary(hf\_quartilerducedModel)



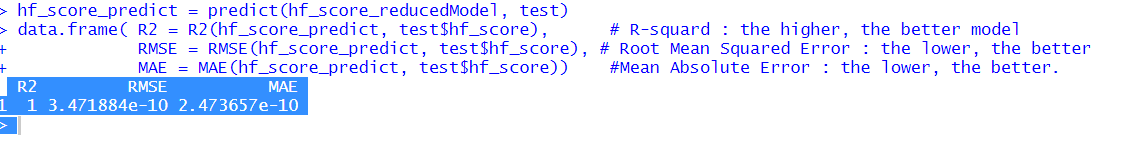
1. Test and conclusion.

hf\_score\_predict = predict(hf\_score\_reducedModel, test)

data.frame( R2 = R2(hf\_score\_predict, test$hf\_score), # R-squard : the higher, the better model

RMSE = RMSE(hf\_score\_predict, test$hf\_score), # Root Mean Squared Error : the lower, the better

MAE = MAE(hf\_score\_predict, test$hf\_score)) #Mean Absolute Error : the lower, the better.

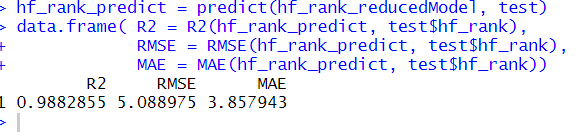


hf\_rank\_predict = predict(hf\_rank\_reducedModel, test)

data.frame( R2 = R2(hf\_rank\_predict, test$hf\_rank),

RMSE = RMSE(hf\_rank\_predict, test$hf\_rank),

MAE = MAE(hf\_rank\_predict, test$hf\_rank))



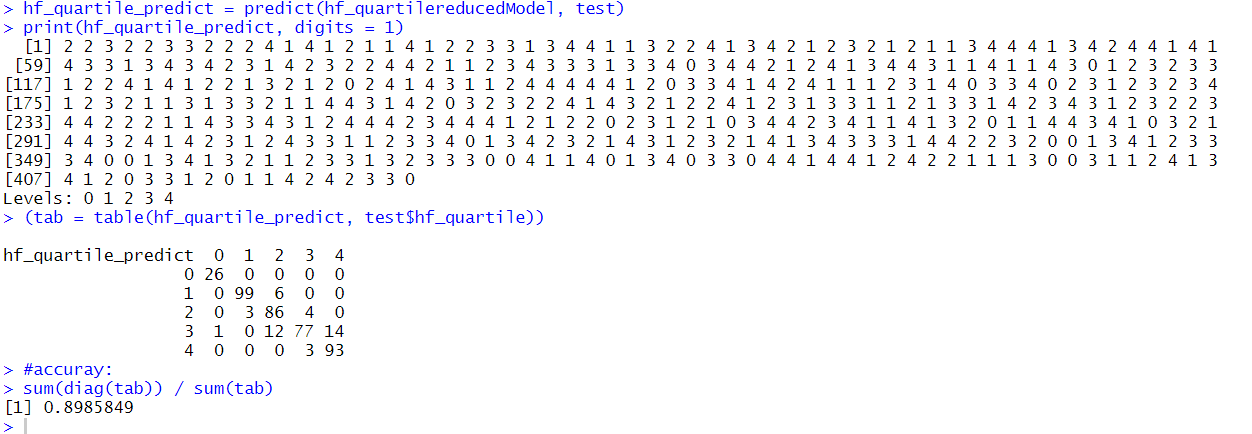
hf\_quartile\_predict = predict(hf\_quartilereducedModel, test)

print(hf\_quartile\_predict, digits = 1)

(tab = table(hf\_quartile\_predict, test$hf\_quartile))

#accuray:

sum(diag(tab)) / sum(tab)



Conclusion

Hf\_score is linearly dependent on **pf\_score** and **ef\_score.**

Hf\_rank is linearly dependent on

**pf\_ss\_homicide , pf\_ss\_disappearances\_violent , pf\_ss\_women , pf\_ss , pf\_movement , pf\_rank , pf\_score , ef\_government\_enterprises , ef\_legal\_integrity , ef\_legal\_restrictions , ef\_money\_sd , ef\_money\_currency, ef\_trade\_black , ef\_regulation\_business\_bureaucracy , ef\_rank**

Hf\_quartile is related to

**pf\_rank , ef\_government , ef\_regulation , ef\_score , ef\_rank**

~~~~~~~ Main.R ~~~~

install.packages('corrplot')

install.packages('caret')

library(corrplot)

library(MASS)

library(tidyverse)

library(caret)

#1.load data

data = read.csv('human-freedom-index-data.csv')

data[is.na(data)] = 0 # NA = 0

#2.data analysis parts- correlation of independent variables. In this step, consider collinearity and minimize data

pf\_rol = data[, c('pf\_rol\_procedural', 'pf\_rol\_civil', 'pf\_rol\_criminal', 'pf\_rol')]

cor(pf\_rol)

corrplot(cor(pf\_rol))

pf\_ss = data[, c('pf\_ss\_homicide', 'pf\_ss\_disappearances\_disap','pf\_ss\_disappearances\_violent','pf\_ss\_disappearances\_organized','pf\_ss\_disappearances\_fatalities','pf\_ss\_disappearances\_injuries','pf\_ss\_disappearances','pf\_ss\_women\_fgm','pf\_ss\_women\_missing','pf\_ss\_women\_inheritance\_widows','pf\_ss\_women\_inheritance\_daughters' ,'pf\_ss\_women\_inheritance','pf\_ss\_women','pf\_ss')]

cor(pf\_ss)

corrplot(cor(pf\_ss))

pf\_movement = data[, c('pf\_movement\_domestic', 'pf\_movement\_foreign','pf\_movement\_women','pf\_movement')]

cor(pf\_movement)

corrplot(cor(pf\_movement))

pf\_religion = data[, c('pf\_religion\_estop\_establish', 'pf\_religion\_estop\_operate','pf\_religion\_estop','pf\_religion\_harassment','pf\_religion\_restrictions','pf\_religion')]

cor(pf\_religion)

corrplot(cor(pf\_religion))

pf\_association = data[, c('pf\_association\_association', 'pf\_association\_assembly','pf\_association\_political\_establish','pf\_association\_political\_operate','pf\_association\_political','pf\_association\_prof\_establish','pf\_association\_prof\_operate','pf\_association\_prof','pf\_association\_sport\_establish')]

cor(pf\_association)

corrplot(cor(pf\_association))

pf\_expression = data[, c('pf\_expression\_killed', 'pf\_expression\_jailed','pf\_expression\_influence','pf\_expression\_control','pf\_expression\_cable','pf\_expression\_newspapers','pf\_expression\_internet','pf\_expression')]

cor(pf\_expression)

corrplot(cor(pf\_expression))

pf\_identity = data[, c('pf\_identity\_legal', 'pf\_identity\_parental\_marriage','pf\_identity\_parental\_divorce','pf\_identity\_parental','pf\_identity\_sex\_male','pf\_identity\_sex\_female','pf\_identity\_sex','pf\_identity\_divorce','pf\_identity')]

corrplot(cor(pf\_identity))

ef\_government = data[, c('ef\_government\_consumption', 'ef\_government\_transfers','ef\_government\_enterprises','ef\_government\_tax\_income','ef\_government\_tax\_payroll','ef\_government\_tax','ef\_government')]

cor(ef\_government)

corrplot(cor(ef\_government))

ef\_legal = data[, c('ef\_legal\_courts', 'ef\_legal\_protection','ef\_legal\_military','ef\_legal\_integrity','ef\_legal\_enforcement','ef\_legal\_restrictions','ef\_legal\_police','ef\_legal\_crime','ef\_legal\_gender','ef\_legal')]

cor(ef\_legal)

corrplot(cor(ef\_legal))

ef\_money = data[, c('ef\_money\_growth', 'ef\_money\_sd','ef\_money\_inflation','ef\_money\_currency','ef\_money')]

cor(ef\_money)

corrplot(cor(ef\_money))

ef\_trade = data[, c('ef\_trade\_tariffs\_revenue', 'ef\_trade\_tariffs\_mean','ef\_trade\_tariffs\_sd','ef\_trade\_tariffs','ef\_trade\_regulatory\_nontariff','ef\_trade\_regulatory\_compliance','ef\_trade\_regulatory','ef\_trade\_black','ef\_trade\_movement\_foreign','ef\_trade\_movement\_capital','ef\_trade\_movement\_visit','ef\_trade\_movement','ef\_trade')]

cor(ef\_trade)

corrplot(cor(ef\_trade))

ef\_regulation = data[, c('ef\_regulation\_business\_adm', 'ef\_regulation\_business\_bureaucracy','ef\_regulation\_business\_start','ef\_regulation\_business\_bribes','ef\_regulation\_business\_licensing','ef\_regulation\_business\_compliance','ef\_regulation\_business','ef\_regulation','ef\_regulation\_credit\_ownership', 'ef\_regulation\_credit\_private','ef\_regulation\_credit\_interest','ef\_regulation\_credit','ef\_regulation\_labor\_minwage','ef\_regulation\_labor\_firing','ef\_regulation\_labor\_bargain','ef\_regulation\_labor\_hours','ef\_regulation\_labor\_dismissal','ef\_regulation\_labor\_conscription','ef\_regulation\_labor')]

cor(ef\_regulation)

corrplot(cor(ef\_regulation))

total = data[, c('hf\_score','hf\_rank','hf\_quartile','pf\_rol','pf\_ss','pf\_movement','pf\_religion','pf\_association','pf\_expression','pf\_identity','pf\_score','pf\_rank','ef\_government','ef\_legal','ef\_money','ef\_trade','ef\_regulation','ef\_score','ef\_rank')]

cor(total)

corrplot(cor(total))

hist(log(data$pf\_rol), main = "Histogram of log(Pf\_rol)", xlab="log(pf\_rol)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_ss), main = "Histogram of log(pf\_ss)", xlab="log(pf\_ss)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_movement), main = "Histogram of log(pf\_movement)", xlab="log(pf\_movement)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_religion), main = "Histogram of log(pf\_religion)", xlab="log(pf\_religion)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_association), main = "Histogram of log(pf\_association)", xlab="log(pf\_association)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_expression), main = "Histogram of log(pf\_expression)", xlab="log(pf\_expression)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_identity), main = "Histogram of log(pf\_identity)", xlab="log(pf\_identity)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_score), main = "Histogram of log(pf\_score)", xlab="log(pf\_score)", ylab="Frequency", freq = TRUE)

hist(log(data$pf\_rank), main = "Histogram of log(pf\_rank)", xlab="log(pf\_rank)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_government), main = "Histogram of log(ef\_government)", xlab="log(ef\_government)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_legal), main = "Histogram of log(ef\_legal)", xlab="log(ef\_legal)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_money), main = "Histogram of log(ef\_money)", xlab="log(ef\_money)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_trade), main = "Histogram of log(ef\_trade)", xlab="log(ef\_trade)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_money), main = "Histogram of log(ef\_money)", xlab="log(ef\_money)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_trade), main = "Histogram of log(ef\_trade)", xlab="log(ef\_trade)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_regulation), main = "Histogram of log(ef\_regulation)", xlab="log(ef\_regulation)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_score), main = "Histogram of log(ef\_score)", xlab="log(ef\_score)", ylab="Frequency", freq = TRUE)

hist(log(data$ef\_rank), main = "Histogram of log(ef\_rank)", xlab="log(ef\_rank)", ylab="Frequency", freq = TRUE)

qqnorm(data$hf\_score); qqline(data$hf\_score)

qqnorm(data$hf\_rank); qqline(data$hf\_rank)

qqnorm(data$hf\_quartile); qqline(data$hf\_quartile)

#data.ggpairs.colnumbers = c()

#ggpairs(data, columns = data.ggpairs.colnumbers)

#minimize data

data['pf\_ss\_women\_inheritance\_widows'] = (data['pf\_ss\_women\_inheritance\_widows'] + data['pf\_ss\_women\_inheritance\_daughters']) / 2

data['pf\_rol\_civil'] = (data['pf\_rol\_procedural'] + data['pf\_rol\_civil'] + data['pf\_rol\_criminal']) / 3

data['pf\_ss\_disappearances'] = (data['pf\_ss\_disappearances'] + data['pf\_ss\_disappearances\_fatalities'] + data['pf\_ss\_disappearances\_injuries'] ) / 3

data['pf\_religion\_estop\_operate'] = (data['pf\_religion\_estop\_operate'] + data['pf\_religion\_estop\_establish']) / 2

data['pf\_religion'] = (data['pf\_religion'] + data['pf\_religion\_harassment']) / 2

data['pf\_association\_association'] = (data['pf\_association\_association'] + data['pf\_association\_assembly']) / 2

data['pf\_expression\_newspapers'] = (data['pf\_expression\_newspapers'] + data['pf\_expression\_cable']) / 2

data['pf\_identity\_parental\_marriage'] = (data['pf\_identity\_parental\_marriage'] + data['pf\_identity\_parental\_divorce']) / 2

data['pf\_identity\_sex'] = (data['pf\_identity\_sex'] + data['pf\_identity\_sex\_male'] + data['pf\_identity\_sex\_female']) / 3

data['ef\_government\_tax'] = (data['ef\_government\_tax'] + data['ef\_government\_tax\_income']) / 2

data['ef\_trade\_regulatory'] = (data['ef\_trade\_regulatory'] + data['ef\_trade\_regulatory\_compliance']) / 2

#3.partition data(train and test = 7:3)

ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))

train <- data[ind==1,]

train <- train[c(names(train)[6:124])]

test <- data[ind==2,]

test <- test[c(names(test)[6:124])]

#4. Results(get Models)

hf\_score\_fullModel = lm(hf\_score ~ .- hf\_rank - hf\_quartile - pf\_ss\_women\_inheritance\_daughters - pf\_rol\_civil - pf\_rol\_criminal - pf\_ss\_disappearances\_injuries - pf\_ss\_disappearances\_fatalities - pf\_religion\_estop\_establish - pf\_religion\_harassment - pf\_association\_assembly - pf\_expression\_cable - pf\_identity\_parental\_divorce - pf\_identity\_sex\_male - pf\_identity\_sex\_female - ef\_government\_tax\_income - ef\_trade\_regulatory\_compliance ,train )

summary(hf\_score\_fullModel)

#final hf\_score model

hf\_score\_reducedModel = lm(hf\_score ~ pf\_score + ef\_score, train)

summary(hf\_score\_reducedModel)

hf\_rank\_fullModel = lm(hf\_rank ~ .- hf\_score - hf\_quartile - pf\_ss\_women\_inheritance\_daughters - pf\_rol\_civil - pf\_rol\_criminal - pf\_ss\_disappearances\_injuries - pf\_ss\_disappearances\_fatalities - pf\_religion\_estop\_establish - pf\_religion\_harassment - pf\_association\_assembly - pf\_expression\_cable - pf\_identity\_parental\_divorce - pf\_identity\_sex\_male - pf\_identity\_sex\_female - ef\_government\_tax\_income - ef\_trade\_regulatory\_compliance ,train )

summary(hf\_rank\_fullModel)

#final hf\_rank model

hf\_rank\_reducedModel = lm(hf\_rank ~ pf\_ss\_homicide + pf\_ss\_disappearances\_violent + pf\_ss\_women + pf\_ss + pf\_movement + pf\_rank + pf\_score + ef\_government\_enterprises + ef\_legal\_integrity + ef\_legal\_restrictions + ef\_money\_sd + ef\_money\_currency + ef\_trade\_black + ef\_regulation\_business\_bureaucracy + ef\_rank, train)

summary(hf\_rank\_reducedModel)

source('fixed-polr.R')

train$hf\_quartile = as.factor(train$hf\_quartile)

hf\_quartilefullModel <- polr(hf\_quartile ~ .- hf\_score - hf\_rank - pf\_ss\_women\_inheritance\_daughters - pf\_rol\_civil - pf\_rol\_criminal - pf\_ss\_disappearances\_injuries - pf\_ss\_disappearances\_fatalities - pf\_religion\_estop\_establish - pf\_religion\_harassment - pf\_association\_assembly - pf\_expression\_cable - pf\_identity\_parental\_divorce - pf\_identity\_sex\_male - pf\_identity\_sex\_female - ef\_government\_tax\_income - ef\_trade\_regulatory\_compliance ,train, Hess = TRUE)

summary(hf\_quartilefullModel)

#p-value calculation

(ctable = coef(summary(hf\_quartilefullModel)))

p = pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) \* 2

(ctabel = cbind(ctable, "p value" = p))

#final hf\_quartile model

hf\_quartilereducedModel <- polr(hf\_quartile ~ pf\_rank + ef\_government + ef\_regulation + ef\_score + ef\_rank, data = train, Hess = TRUE )

summary(hf\_quartilereducedModel)

(ctable = coef(summary(hf\_quartilereducedModel)))

p = pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) \* 2

(ctabel = cbind(ctable, "p value" = p))

#5. Testing the models.

# normal Test method.

hf\_score\_predict = predict(hf\_score\_reducedModel, test)

data.frame( R2 = R2(hf\_score\_predict, test$hf\_score), # R-squard : the higher, the better model

RMSE = RMSE(hf\_score\_predict, test$hf\_score), # Root Mean Squared Error : the lower, the better

MAE = MAE(hf\_score\_predict, test$hf\_score)) #Mean Absolute Error : the lower, the better.

hf\_rank\_predict = predict(hf\_rank\_reducedModel, test)

data.frame( R2 = R2(hf\_rank\_predict, test$hf\_rank),

RMSE = RMSE(hf\_rank\_predict, test$hf\_rank),

MAE = MAE(hf\_rank\_predict, test$hf\_rank))

hf\_quartile\_predict = predict(hf\_quartilereducedModel, test)

print(hf\_quartile\_predict, digits = 1)

(tab = table(hf\_quartile\_predict, test$hf\_quartile))

#accuray:

sum(diag(tab)) / sum(tab)

# Cross Validation Test

set.seed(123)

train.control = trainControl(method = "cv", number = 10)

hf\_scoreModel = train(hf\_score ~ pf\_score + ef\_score + ef\_regulation\_business\_start, data = data, method = "lm", trControl = train.control)

print(hf\_scoreModel)

train.control = trainControl(method = "cv", number = 10)

hf\_rankModel = train(hf\_rank ~ pf\_ss\_homicide + pf\_ss\_disappearances\_violent + pf\_ss\_women + pf\_ss + pf\_movement + pf\_religion\_estop\_operate + pf\_religion\_estop + pf\_association\_political\_establish + pf\_association\_political + pf\_expression\_jailed + pf\_identity\_sex + pf\_rank + pf\_score + ef\_government\_enterprises + ef\_legal\_integrity + ef\_legal\_restrictions + ef\_money\_sd + ef\_money\_currency + ef\_trade\_black + ef\_regulation\_business\_bureaucracy + ef\_rank, data = data, method = "lm", trControl = train.control)

print(hf\_rankModel)

~~~~~~~~ fixed\_polr.R ~~~~~~~~~~

polr <- function(formula, data, weights, start, ..., subset,

na.action, contrasts = NULL, Hess = FALSE,

model = TRUE,

method = c("logistic", "probit", "cloglog", "cauchit"))

{

logit <- function(p) log(p/(1 - p))

fmin <- function(beta) {

theta <- beta[pc + 1L:q]

gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)

eta <- offset

if (pc > 0)

eta <- eta + drop(x %\*% beta[1L:pc])

pr <- pfun(gamm[y + 1] - eta) - pfun(gamm[y] - eta)

if (all(pr > 0))

-sum(wt \* log(pr))

else Inf

}

gmin <- function(beta)

{

jacobian <- function(theta) { ## dgamma by dtheta matrix

k <- length(theta)

etheta <- exp(theta)

mat <- matrix(0 , k, k)

mat[, 1] <- rep(1, k)

for (i in 2:k) mat[i:k, i] <- etheta[i]

mat

}

theta <- beta[pc + 1L:q]

gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)

eta <- offset

if(pc > 0) eta <- eta + drop(x %\*% beta[1L:pc])

pr <- pfun(gamm[y+1] - eta) - pfun(gamm[y] - eta)

p1 <- dfun(gamm[y+1] - eta)

p2 <- dfun(gamm[y] - eta)

g1 <- if(pc > 0) t(x) %\*% (wt\*(p1 - p2)/pr) else numeric(0)

xx <- .polrY1\*p1 - .polrY2\*p2

g2 <- - t(xx) %\*% (wt/pr)

g2 <- t(g2) %\*% jacobian(theta)

if(all(pr > 0)) c(g1, g2) else rep(NA, pc+q)

}

m <- match.call(expand.dots = FALSE)

method <- match.arg(method)

pfun <- switch(method, logistic = plogis, probit = pnorm,

cloglog = pgumbel, cauchit = pcauchy)

dfun <- switch(method, logistic = dlogis, probit = dnorm,

cloglog = dgumbel, cauchit = dcauchy)

if(is.matrix(eval.parent(m$data)))

m$data <- as.data.frame(data)

m$start <- m$Hess <- m$method <- m$model <- m$... <- NULL

m[[1L]] <- as.name("model.frame")

m <- eval.parent(m)

Terms <- attr(m, "terms")

x <- model.matrix(Terms, m, contrasts)

xint <- match("(Intercept)", colnames(x), nomatch=0L)

n <- nrow(x)

pc <- ncol(x)

cons <- attr(x, "contrasts") # will get dropped by subsetting

if(xint > 0) {

x <- x[, -xint, drop=FALSE]

pc <- pc - 1

} else warning("an intercept is needed and assumed")

wt <- model.weights(m)

if(!length(wt)) wt <- rep(1, n)

offset <- model.offset(m)

if(length(offset) <= 1) offset <- rep(0, n)

y <- model.response(m)

if(!is.factor(y)) stop("response must be a factor")

lev <- levels(y)

if(length(lev) <= 2) stop("response must have 3 or more levels")

y <- unclass(y)

q <- length(lev) - 1

Y <- matrix(0, n, q)

.polrY1 <- col(Y) == y

.polrY2 <- col(Y) == y - 1

if(missing(start)) {

# try something that should always work -tjb

u <- as.integer(table(y))

u <- (cumsum(u)/sum(u))[1:q]

zetas <-

switch(method,

"logistic"= qlogis(u),

"probit"= qnorm(u),

"cauchit"= qcauchy(u),

"cloglog"= -log(-log(u)) )

s0 <- c(rep(0,pc),zetas[1],log(diff(zetas)))

## # try logistic/probit regression on 'middle' cut

## q1 <- length(lev) %/% 2

## y1 <- (y > q1)

## X <- cbind(Intercept = rep(1, n), x)

## fit <-

## switch(method,

## "logistic"= glm.fit(X, y1, wt, family = binomial(), offset = offset),

## "probit" = glm.fit(X, y1, wt, family = binomial("probit"), offset = offset),

## ## this is deliberate, a better starting point

## "cloglog" = glm.fit(X, y1, wt, family = binomial("probit"), offset = offset),

## "cauchit" = glm.fit(X, y1, wt, family = binomial("cauchit"), offset = offset))

## if(!fit$converged)

## stop("attempt to find suitable starting values failed")

## coefs <- fit$coefficients

## if(any(is.na(coefs))) {

## warning("design appears to be rank-deficient, so dropping some coefs")

## keep <- names(coefs)[!is.na(coefs)]

## coefs <- coefs[keep]

## x <- x[, keep[-1L], drop = FALSE]

## pc <- ncol(x)

## }

## spacing <- logit((1L:q)/(q+1)) # just a guess

## if(method != "logistic") spacing <- spacing/1.7

## gammas <- -coefs[1L] + spacing - spacing[q1]

## thetas <- c(gammas[1L], log(diff(gammas)))

## s0 <- c(coefs[-1L], thetas)

} else if(length(start) != pc + q)

stop("'start' is not of the correct length")

else {

s0 <- if(pc > 0) c(start[seq\_len(pc+1)], log(diff(start[-seq\_len(pc)])))

else c(start[1L], log(diff(start)))

}

res <- optim(s0, fmin, gmin, method="BFGS", hessian = Hess, ...)

beta <- res$par[seq\_len(pc)]

theta <- res$par[pc + 1L:q]

zeta <- cumsum(c(theta[1L],exp(theta[-1L])))

deviance <- 2 \* res$value

niter <- c(f.evals=res$counts[1L], g.evals=res$counts[2L])

names(zeta) <- paste(lev[-length(lev)], lev[-1L], sep="|")

if(pc > 0) {

names(beta) <- colnames(x)

eta <- offset + drop(x %\*% beta)

} else eta <- offset + rep(0, n)

cumpr <- matrix(pfun(matrix(zeta, n, q, byrow=TRUE) - eta), , q)

fitted <- t(apply(cumpr, 1L, function(x) diff(c(0, x, 1))))

dimnames(fitted) <- list(row.names(m), lev)

fit <- list(coefficients = beta, zeta = zeta, deviance = deviance,

fitted.values = fitted, lev = lev, terms = Terms,

df.residual = sum(wt) - pc - q, edf = pc + q, n = sum(wt),

nobs = sum(wt),

call = match.call(), method = method,

convergence = res$convergence, niter = niter, lp = eta)

if(Hess) {

dn <- c(names(beta), names(zeta))

H <- res$hessian

dimnames(H) <- list(dn, dn)

fit$Hessian <- H

}

if(model) fit$model <- m

fit$na.action <- attr(m, "na.action")

fit$contrasts <- cons

fit$xlevels <- .getXlevels(Terms, m)

class(fit) <- "polr"

fit

}

print.polr <- function(x, ...)

{

if(!is.null(cl <- x$call)) {

cat("Call:\n")

dput(cl, control=NULL)

}

if(length(coef(x))) {

cat("\nCoefficients:\n")

print(coef(x), ...)

} else {

cat("\nNo coefficients\n")

}

cat("\nIntercepts:\n")

print(x$zeta, ...)

cat("\nResidual Deviance:", format(x$deviance, nsmall=2), "\n")

cat("AIC:", format(x$deviance + 2\*x$edf, nsmall=2), "\n")

if(nzchar(mess <- naprint(x$na.action))) cat("(", mess, ")\n", sep="")

if(x$convergence > 0)

cat("Warning: did not converge as iteration limit reached\n")

invisible(x)

}

vcov.polr <- function(object, ...)

{

jacobian <- function(theta) { ## dgamma by dtheta matrix

k <- length(theta)

etheta <- exp(theta)

mat <- matrix(0 , k, k)

mat[, 1] <- rep(1, k)

for (i in 2:k) mat[i:k, i] <- etheta[i]

mat

}

if(is.null(object$Hessian)) {

message("\nRe-fitting to get Hessian\n")

utils::flush.console()

object <- update(object, Hess=TRUE,

start=c(object$coefficients, object$zeta))

}

vc <- ginv(object$Hessian)

pc <- length(coef(object))

gamma <- object$zeta

z.ind <- pc + seq\_along(gamma)

theta <- c(gamma[1L], log(diff(gamma)))

J <- jacobian(theta)

A <- diag(pc + length(gamma))

A[z.ind, z.ind] <- J

V <- A %\*% vc %\*% t(A)

structure(V, dimnames = dimnames(object$Hessian))

}

summary.polr <- function(object, digits = max(3, .Options$digits - 3),

correlation = FALSE, ...)

{

cc <- c(coef(object), object$zeta)

pc <- length(coef(object))

q <- length(object$zeta)

coef <- matrix(0, pc+q, 3, dimnames=list(names(cc),

c("Value", "Std. Error", "t value")))

coef[, 1] <- cc

vc <- vcov(object)

coef[, 2] <- sd <- sqrt(diag(vc))

coef[, 3] <- coef[, 1]/coef[, 2]

object$coefficients <- coef

object$pc <- pc

object$digits <- digits

if(correlation)

object$correlation <- (vc/sd)/rep(sd, rep(pc+q, pc+q))

class(object) <- "summary.polr"

object

}

print.summary.polr <- function(x, digits = x$digits, ...)

{

if(!is.null(cl <- x$call)) {

cat("Call:\n")

dput(cl, control=NULL)

}

coef <- format(round(x$coefficients, digits=digits))

pc <- x$pc

if(pc > 0) {

cat("\nCoefficients:\n")

print(x$coefficients[seq\_len(pc), , drop=FALSE], quote = FALSE,

digits = digits, ...)

} else {

cat("\nNo coefficients\n")

}

cat("\nIntercepts:\n")

print(coef[(pc+1):nrow(coef), , drop=FALSE], quote = FALSE,

digits = digits, ...)

cat("\nResidual Deviance:", format(x$deviance, nsmall=2), "\n")

cat("AIC:", format(x$deviance + 2\*x$edf, nsmall=2), "\n")

if(nzchar(mess <- naprint(x$na.action))) cat("(", mess, ")\n", sep="")

if(!is.null(correl <- x$correlation)) {

cat("\nCorrelation of Coefficients:\n")

ll <- lower.tri(correl)

correl[ll] <- format(round(correl[ll], digits))

correl[!ll] <- ""

print(correl[-1, -ncol(correl)], quote = FALSE, ...)

}

invisible(x)

}

predict.polr <- function(object, newdata, type=c("class","probs"), ...)

{

if(!inherits(object, "polr")) stop("not a \"polr\" object")

type <- match.arg(type)

if(missing(newdata)) Y <- object$fitted

else {

newdata <- as.data.frame(newdata)

Terms <- delete.response(object$terms)

m <- model.frame(Terms, newdata, na.action = function(x) x,

xlev = object$xlevels)

if (!is.null(cl <- attr(Terms, "dataClasses")))

.checkMFClasses(cl, m)

X <- model.matrix(Terms, m, contrasts = object$contrasts)

xint <- match("(Intercept)", colnames(X), nomatch=0L)

if(xint > 0) X <- X[, -xint, drop=FALSE]

n <- nrow(X)

q <- length(object$zeta)

eta <- drop(X %\*% object$coefficients)

pfun <- switch(object$method, logistic = plogis, probit = pnorm,

cloglog = pgumbel, cauchit = pcauchy)

cumpr <- matrix(pfun(matrix(object$zeta, n, q, byrow=TRUE) - eta), , q)

Y <- t(apply(cumpr, 1L, function(x) diff(c(0, x, 1))))

dimnames(Y) <- list(rownames(X), object$lev)

}

if(missing(newdata) && !is.null(object$na.action))

Y <- napredict(object$na.action, Y)

switch(type, class = {

Y <- factor(max.col(Y), levels=seq\_along(object$lev),

labels=object$lev)

}, probs = {})

drop(Y)

}

extractAIC.polr <- function(fit, scale = 0, k = 2, ...)

{

edf <- fit$edf

c(edf, deviance(fit) + k \* edf)

}

model.frame.polr <- function(formula, ...)

{

dots <- list(...)

nargs <- dots[match(c("data", "na.action", "subset"), names(dots), 0)]

if(length(nargs) || is.null(formula$model)) {

m <- formula$call

m$start <- m$Hess <- m$... <- NULL

m[[1L]] <- as.name("model.frame")

m[names(nargs)] <- nargs

if (is.null(env <- environment(formula$terms))) env <- parent.frame()

data <- eval(m, env)

if(!is.null(mw <- m$weights)) {

nm <- names(data)

nm[match("(weights)", nm)] <- as.character(mw)

names(data) <- nm

}

data

} else formula$model

}

pgumbel <- function(q, loc = 0, scale = 1, lower.tail = TRUE)

{

q <- (q - loc)/scale

p <- exp(-exp(-q))

if (!lower.tail) 1 - p else p

}

dgumbel <- function (x, loc = 0, scale = 1, log = FALSE)

{

x <- (x - loc)/scale

d <- log(1/scale) - x - exp(-x)

d[is.nan(d)] <- -Inf # -tjb

if (!log) exp(d) else d

}

anova.polr <- function (object, ..., test = c("Chisq", "none"))

{

test <- match.arg(test)

dots <- list(...)

if (length(dots) == 0L)

stop('anova is not implemented for a single "polr" object')

mlist <- list(object, ...)

nt <- length(mlist)

dflis <- sapply(mlist, function(x) x$df.residual)

s <- order(dflis, decreasing = TRUE)

mlist <- mlist[s]

if (any(!sapply(mlist, inherits, "polr")))

stop('not all objects are of class "polr"')

ns <- sapply(mlist, function(x) length(x$fitted.values))

if(any(ns != ns[1L]))

stop("models were not all fitted to the same size of dataset")

rsp <- unique(sapply(mlist, function(x) paste(formula(x)[2L])))

mds <- sapply(mlist, function(x) paste(formula(x)[3L]))

dfs <- dflis[s]

lls <- sapply(mlist, function(x) deviance(x))

tss <- c("", paste(1L:(nt - 1), 2:nt, sep = " vs "))

df <- c(NA, -diff(dfs))

x2 <- c(NA, -diff(lls))

pr <- c(NA, 1 - pchisq(x2[-1L], df[-1L]))

out <- data.frame(Model = mds, Resid.df = dfs, Deviance = lls,

Test = tss, Df = df, LRtest = x2, Prob = pr)

names(out) <- c("Model", "Resid. df", "Resid. Dev", "Test",

" Df", "LR stat.", "Pr(Chi)")

if (test == "none") out <- out[, 1L:6]

class(out) <- c("Anova", "data.frame")

attr(out, "heading") <-

c("Likelihood ratio tests of ordinal regression models\n",

paste("Response:", rsp))

out

}

polr.fit <- function(x, y, wt, start, offset, method)

{

logit <- function(p) log(p/(1 - p))

fmin <- function(beta) {

theta <- beta[pc + 1L:q]

gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)

eta <- offset

if (pc > 0)

eta <- eta + drop(x %\*% beta[1L:pc])

pr <- pfun(gamm[y + 1] - eta) - pfun(gamm[y] - eta)

if (all(pr > 0))

-sum(wt \* log(pr))

else Inf

}

gmin <- function(beta)

{

jacobian <- function(theta) { ## dgamma by dtheta matrix

k <- length(theta)

etheta <- exp(theta)

mat <- matrix(0 , k, k)

mat[, 1] <- rep(1, k)

for (i in 2:k) mat[i:k, i] <- etheta[i]

mat

}

theta <- beta[pc + 1L:q]

gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)

eta <- offset

if(pc > 0) eta <- eta + drop(x %\*% beta[1L:pc])

pr <- pfun(gamm[y+1] - eta) - pfun(gamm[y] - eta)

p1 <- dfun(gamm[y+1] - eta)

p2 <- dfun(gamm[y] - eta)

g1 <- if(pc > 0) t(x) %\*% (wt\*(p1 - p2)/pr) else numeric(0)

xx <- .polrY1\*p1 - .polrY2\*p2

g2 <- - t(xx) %\*% (wt/pr)

g2 <- t(g2) %\*% jacobian(theta)

if(all(pr > 0)) c(g1, g2) else rep(NA, pc+q)

}

pfun <- switch(method, logistic = plogis, probit = pnorm,

cloglog = pgumbel, cauchit = pcauchy)

dfun <- switch(method, logistic = dlogis, probit = dnorm,

cloglog = dgumbel, cauchit = dcauchy)

n <- nrow(x)

pc <- ncol(x)

lev <- levels(y)

if(length(lev) <= 2L) stop("response must have 3 or more levels")

y <- unclass(y)

q <- length(lev) - 1L

Y <- matrix(0, n, q)

.polrY1 <- col(Y) == y

.polrY2 <- col(Y) == y - 1L

# pc could be 0.

s0 <- if(pc > 0) c(start[seq\_len(pc+1)], diff(start[-seq\_len(pc)]))

else c(start[1L], diff(start))

res <- optim(s0, fmin, gmin, method="BFGS")

beta <- res$par[seq\_len(pc)]

theta <- res$par[pc + 1L:q]

zeta <- cumsum(c(theta[1L],exp(theta[-1L])))

deviance <- 2 \* res$value

names(zeta) <- paste(lev[-length(lev)], lev[-1L], sep="|")

if(pc > 0) {

names(beta) <- colnames(x)

eta <- drop(x %\*% beta)

} else {

eta <- rep(0, n)

}

list(coefficients = beta, zeta = zeta, deviance = deviance)

}

profile.polr <- function(fitted, which = 1L:p, alpha = 0.01,

maxsteps = 10, del = zmax/5, trace = FALSE, ...)

{

Pnames <- names(B0 <- coefficients(fitted))

pv0 <- t(as.matrix(B0))

p <- length(Pnames)

if(is.character(which)) which <- match(which, Pnames)

summ <- summary(fitted)

std.err <- summ$coefficients[, "Std. Error"]

mf <- model.frame(fitted)

n <- length(Y <- model.response(mf))

O <- model.offset(mf)

if(!length(O)) O <- rep(0, n)

W <- model.weights(mf)

if(length(W) == 0L) W <- rep(1, n)

OriginalDeviance <- deviance(fitted)

X <- model.matrix(fitted)[, -1L, drop=FALSE] # drop intercept

zmax <- sqrt(qchisq(1 - alpha, 1))

profName <- "z"

prof <- vector("list", length=length(which))

names(prof) <- Pnames[which]

start <- c(fitted$coefficients, fitted$zeta)

for(i in which) {

zi <- 0

pvi <- pv0

Xi <- X[, - i, drop = FALSE]

pi <- Pnames[i]

for(sgn in c(-1, 1)) {

if(trace) {

message("\nParameter:", pi, c("down", "up")[(sgn + 1)/2 + 1])

utils::flush.console()

}

step <- 0

z <- 0

## LP is the linear predictor including offset.

## LP <- X %\*% fitted$coef + O

while((step <- step + 1) < maxsteps && abs(z) < zmax) {

bi <- B0[i] + sgn \* step \* del \* std.err[i]

o <- O + X[, i] \* bi

fm <- polr.fit(x = Xi, y = Y, wt = W, start = start[-i],

offset = o, method = fitted$method)

ri <- pv0

ri[, names(coef(fm))] <- coef(fm)

ri[, pi] <- bi

pvi <- rbind(pvi, ri)

zz <- fm$deviance - OriginalDeviance

if(zz > - 1e-3) zz <- max(zz, 0)

else stop("profiling has found a better solution, so original fit had not converged")

z <- sgn \* sqrt(zz)

zi <- c(zi, z)

}

}

si <- order(zi)

prof[[pi]] <- structure(data.frame(zi[si]), names = profName)

prof[[pi]]$par.vals <- pvi[si, ]

}

val <- structure(prof, original.fit = fitted, summary = summ)

class(val) <- c("profile.polr", "profile")

val

}

confint.polr <- function(object, parm, level = 0.95, trace = FALSE, ...)

{

pnames <- names(coef(object))

if(missing(parm)) parm <- seq\_along(pnames)

else if(is.character(parm)) parm <- match(parm, pnames, nomatch = 0L)

message("Waiting for profiling to be done...")

utils::flush.console()

object <- profile(object, which = parm, alpha = (1. - level)/4.,

trace = trace)

confint(object, parm=parm, level=level, trace=trace, ...)

}

confint.profile.polr <-

function(object, parm = seq\_along(pnames), level = 0.95, ...)

{

of <- attr(object, "original.fit")

pnames <- names(coef(of))

if(is.character(parm)) parm <- match(parm, pnames, nomatch = 0L)

a <- (1-level)/2

a <- c(a, 1-a)

pct <- paste(round(100\*a, 1), "%")

ci <- array(NA, dim = c(length(parm), 2L),

dimnames = list(pnames[parm], pct))

cutoff <- qnorm(a)

for(pm in parm) {

pro <- object[[ pnames[pm] ]]

if(length(pnames) > 1L)

sp <- spline(x = pro[, "par.vals"][, pm], y = pro[, 1])

else sp <- spline(x = pro[, "par.vals"], y = pro[, 1])

ci[pnames[pm], ] <- approx(sp$y, sp$x, xout = cutoff)$y

}

drop(ci)

}

logLik.polr <- function(object, ...)

structure(-0.5 \* object$deviance, df = object$edf, class = "logLik")

simulate.polr <- function(object, nsim = 1, seed = NULL, ...)

{

if(!is.null(object$model) && any(model.weights(object$model) != 1))

stop("weighted fits are not supported")

rgumbel <- function(n, loc = 0, scale = 1) loc - scale\*log(rexp(n))

## start the same way as simulate.lm

if(!exists(".Random.seed", envir = .GlobalEnv, inherits = FALSE))

runif(1) # initialize the RNG if necessary

if(is.null(seed))

RNGstate <- get(".Random.seed", envir = .GlobalEnv)

else {

R.seed <- get(".Random.seed", envir = .GlobalEnv)

set.seed(seed)

RNGstate <- structure(seed, kind = as.list(RNGkind()))

on.exit(assign(".Random.seed", R.seed, envir = .GlobalEnv))

}

rfun <- switch(object$method, logistic = rlogis, probit = rnorm,

cloglog = rgumbel, cauchit = rcauchy)

eta <- object$lp

n <- length(eta)

res <- cut(rfun(n\*nsim, eta),

c(-Inf, object$zeta, Inf),

labels = colnames(fitted(object)),

ordered\_result = TRUE)

val <- split(res, rep(seq\_len(nsim), each=n))

names(val) <- paste("sim", seq\_len(nsim), sep="\_")

val <- as.data.frame(val)

if (!is.null(nm <- rownames(fitted(object)))) row.names(val) <- nm

attr(val, "seed") <- RNGstate

val

}

Thank you…

2019.8.30 RichardBill