**[ 카이제곱 검정 모델 ]**

library(rhdfs)

hdfs.init()

library(rmr2)

rmr.options("hadoop")

# 폴더에 포함된 파일 목록 files에 할당

files <- hdfs.ls("/data/taxi/combined")$file

info.format <- make.input.format( format = "csv",

sep = ",",

colClasses = "character",

stringsAsFactors = FALSE )

res <- from.dfs(input = files[1], format = info.format)

info <- values(res)

colnames.tmp <- as.character(info[,1])

colclass.tmp <- as.character(info[,2])

colnames <- colnames.tmp[-1]

colclasses <- colclass.tmp[-1]

colclasses[c(6,8,9,10)] <- "numeric"

taxi.format <- make.input.format(format = "csv", sep = ",",

col.names = colnames,

colClasses = colclasses,

stringsAsFactors = FALSE)

files <- files[-1]; files

# 2013년 뉴욕 불쾌지수 데이터 가져오기

discomfort\_csv <- read.csv("month\_day\_num.csv");

dc1 <- as.numeric(discomfort\_csv[1,])

dc2 <- as.numeric(discomfort\_csv[2,])

dc3 <- as.numeric(discomfort\_csv[3,])

dc4 <- as.numeric(discomfort\_csv[4,])

dc5 <- as.numeric(discomfort\_csv[5,])

dc6 <- as.numeric(discomfort\_csv[6,])

dc7 <- as.numeric(discomfort\_csv[7,])

dc8 <- as.numeric(discomfort\_csv[8,])

dc9 <- as.numeric(discomfort\_csv[9,])

dc10 <- as.numeric(discomfort\_csv[10,])

dc11 <- as.numeric(discomfort\_csv[11,])

dc12 <- as.numeric(discomfort\_csv[12,])

## 카이제곱 map 함수

chi.map.fun <- function(k, v){

taxi <- v

taxi <- taxi[taxi$payment\_type=="CRD",] # payment\_type == CRD

taxi <- taxi[(taxi$fare\_amount>=2.5),] # fare\_amount >= 2.5

taxi <- taxi[(taxi$rate\_code==1)|(taxi$rate\_code==2)|(taxi$rate\_code==3)|(taxi$rate\_code==4)|(taxi$rate\_code==5)|(taxi$rate\_code==6),] # rate\_code == 1,2,3,4,5,6

taxi <- taxi[(taxi$passenger\_count>=1),] # passenger\_count >= 1

taxi <- taxi[(taxi$trip\_time\_in\_secs>0&taxi$trip\_time\_in\_secs<10000),] # 0 < trip\_time\_in\_secs < 10000

taxi <- taxi[(taxi$trip\_distance>0) & (taxi$trip\_distance<200),] # 0 < trip\_distance < 200

taxi <- taxi[(taxi$pickup\_longitude>(-80)) & (taxi$pickup\_longitude<(-70)),] # -80 < pickup\_longitude < -70

taxi <- taxi[(taxi$pickup\_latitude>40) & (taxi$pickup\_latitude<46),] # 40 < pickup\_latitude < 46

taxi <- taxi[(taxi$dropoff\_longitude>(-80)) & (taxi$dropoff\_longitude<(-70)),] # -80 < dropoff\_longitude < -70

taxi <- taxi[(taxi$dropoff\_latitude>40) & (taxi$dropoff\_latitude<46),] # 40 < dropoff\_latitude < 46

taxi$discomfort <- -1

month <- as.integer(substr(taxi$pickup\_datetime, 6, 7))

date <- as.integer(substr(taxi$pickup\_datetime, 9, 10))

taxi$pickup\_month <- month

taxi$pickup\_date <- date

# 불쾌지수 뽑아서 넣기

ifelse(month == 1, taxi$discomfort <- dc1[date], NA)

ifelse(month == 2, taxi$discomfort <- dc2[date], NA)

ifelse(month == 3, taxi$discomfort <- dc3[date], NA)

ifelse(month == 4, taxi$discomfort <- dc4[date], NA)

ifelse(month == 5, taxi$discomfort <- dc5[date], NA)

ifelse(month == 6, taxi$discomfort <- dc6[date], NA)

ifelse(month == 7, taxi$discomfort <- dc7[date], NA)

ifelse(month == 8, taxi$discomfort <- dc8[date], NA)

ifelse(month == 9, taxi$discomfort <- dc9[date], NA)

ifelse(month == 10, taxi$discomfort <- dc10[date], NA)

ifelse(month == 11, taxi$discomfort <- dc11[date], NA)

ifelse(month == 12, taxi$discomfort <- dc12[date], NA)

tip\_ratio <- taxi$tip\_amount / taxi$total\_amount

taxi$tip\_ratio <- tip\_ratio

taxi$tip\_ratio\_index <- 1

#ifelse(tip\_ratio < 0.1, dat9$tip\_ratio\_index <- 0, dat9$tip\_ratio\_index <- 1)

ind10 <- ((taxi$tip\_amount / taxi$total\_amount) < 0.1)

taxi[ind10,]$tip\_ratio\_index <- 0

#ifelse(tip\_ratio > 0.2, dat9$tip\_ratio\_index <- 2, NA)

ind11 <- ((taxi$tip\_amount / taxi$total\_amount) > 0.2)

taxi[ind11,]$tip\_ratio\_index <- 2

taxi

}

chi.mr <- mapreduce( input = files,

input.format = taxi.format,

map = chi.map.fun)

chi.res <- from.dfs(mr)

## 이원 카이제곱 검정

# 각 집단 당 개수를 100만개로 맞추기

data <- res$val

data0 <- data[data$discomfort == 0,]

data1 <- data[data$discomfort == 1,]

data2 <- data[data$discomfort == 2,]

set.seed(123)

t0 <- sample(nrow(data0), 1000000, replace = F)

t1 <- sample(nrow(data1), 1000000, replace = F)

t2 <- sample(nrow(data2), 1000000, replace = F)

data0 <- data0[t0,]

data1 <- data1[t1,]

data2 <- data2[t2,]

total <- rbind(data0, data1)

total <- rbind(total, data2)

# 변수 리코딩

total$discomfort2[total$discomfort == 0] <- "3. 불쾌"

total$discomfort2[total$discomfort == 1] <- "2. 중간"

total$discomfort2[total$discomfort == 2] <- "1. 상쾌"

total$tip\_ratio\_index2[total$tip\_ratio\_index == 0] <- "1. 이하"

total$tip\_ratio\_index2[total$tip\_ratio\_index == 1] <- "2. 적정(10~20%)"

total$tip\_ratio\_index2[total$tip\_ratio\_index == 2] <- "3. 이상"

# 교차 분할표 작성

table(total$discomfort2, total$tip\_ratio\_index2)

# 동질성 검사 - 모두 특성치에 대한 추론검정

chisq.test(total$discomfort2, total$tip\_ratio\_index2)

**[ 회귀 분석 모델 ]**

library(rhdfs)

hdfs.init()

library(rmr2)

rmr.options("hadoop")

# 폴더에 포함된 파일 목록 files에 할당

files <- hdfs.ls("/data/taxi/combined")$file

info.format <- make.input.format( format = "csv",

sep = ",",

colClasses = "character",

stringsAsFactors = FALSE )

res <- from.dfs(input = files[1], format = info.format)

info <- values(res)

colnames.tmp <- as.character(info[,1])

colclass.tmp <- as.character(info[,2])

colnames <- colnames.tmp[-1]

colclasses <- colclass.tmp[-1]

colclasses[c(6,8,9,10)] <- "numeric"

taxi.format <- make.input.format(format = "csv", sep = ",",

col.names = colnames,

colClasses = colclasses,

stringsAsFactors = FALSE)

files <- files[-1]; files

# 2013년 뉴욕 불쾌지수 데이터 가져오기

discomfort\_csv <- read.csv("month\_day\_num.csv");

dc1 <- as.numeric(discomfort\_csv[1,])

dc2 <- as.numeric(discomfort\_csv[2,])

dc3 <- as.numeric(discomfort\_csv[3,])

dc4 <- as.numeric(discomfort\_csv[4,])

dc5 <- as.numeric(discomfort\_csv[5,])

dc6 <- as.numeric(discomfort\_csv[6,])

dc7 <- as.numeric(discomfort\_csv[7,])

dc8 <- as.numeric(discomfort\_csv[8,])

dc9 <- as.numeric(discomfort\_csv[9,])

dc10 <- as.numeric(discomfort\_csv[10,])

dc11 <- as.numeric(discomfort\_csv[11,])

dc12 <- as.numeric(discomfort\_csv[12,])

## 회귀분석 map 함수

reg.map.fun <- function(k, v){

taxi <- v

taxi <- taxi[taxi$payment\_type=="CRD",] # payment\_type == CRD

taxi <- taxi[(taxi$fare\_amount>=2.5),] # fare\_amount >= 2.5

taxi <- taxi[(taxi$rate\_code==1)|(taxi$rate\_code==2)|(taxi$rate\_code==3)|(taxi$rate\_code==4)|(taxi$rate\_code==5)|(taxi$rate\_code==6),] # rate\_code == 1,2,3,4,5,6

taxi <- taxi[(taxi$passenger\_count>=1),] # passenger\_count >= 1

taxi <- taxi[(taxi$trip\_time\_in\_secs>0&taxi$trip\_time\_in\_secs<10000),] # 0 < trip\_time\_in\_secs < 10000

taxi <- taxi[(taxi$trip\_distance>0) & (taxi$trip\_distance<200),] # 0 < trip\_distance < 200

taxi <- taxi[(taxi$pickup\_longitude>(-80)) & (taxi$pickup\_longitude<(-70)),] # -80 < pickup\_longitude < -70

taxi <- taxi[(taxi$pickup\_latitude>40) & (taxi$pickup\_latitude<46),] # 40 < pickup\_latitude < 46

taxi <- taxi[(taxi$dropoff\_longitude>(-80)) & (taxi$dropoff\_longitude<(-70)),] # -80 < dropoff\_longitude < -70

taxi <- taxi[(taxi$dropoff\_latitude>40) & (taxi$dropoff\_latitude<46),] # 40 < dropoff\_latitude < 46

taxi$discomfort <- -1

month <- as.integer(substr(taxi$pickup\_datetime, 6, 7))

date <- as.integer(substr(taxi$pickup\_datetime, 9, 10))

taxi$pickup\_month <- month

taxi$pickup\_date <- date

# 불쾌지수 뽑아서 넣기

ifelse(month == 1, taxi$discomfort <- dc1[date], NA)

ifelse(month == 2, taxi$discomfort <- dc2[date], NA)

ifelse(month == 3, taxi$discomfort <- dc3[date], NA)

ifelse(month == 4, taxi$discomfort <- dc4[date], NA)

ifelse(month == 5, taxi$discomfort <- dc5[date], NA)

ifelse(month == 6, taxi$discomfort <- dc6[date], NA)

ifelse(month == 7, taxi$discomfort <- dc7[date], NA)

ifelse(month == 8, taxi$discomfort <- dc8[date], NA)

ifelse(month == 9, taxi$discomfort <- dc9[date], NA)

ifelse(month == 10, taxi$discomfort <- dc10[date], NA)

ifelse(month == 11, taxi$discomfort <- dc11[date], NA)

ifelse(month == 12, taxi$discomfort <- dc12[date], NA)

tip\_ratio <- taxi$tip\_amount / taxi$total\_amount

taxi$tip\_ratio <- tip\_ratio

### 1

## discomfort index model

dat <- data.frame(tip\_ratio = taxi$tip\_ratio,discomfort = taxi$discomfort)

tip <- dat$tip\_ratio

discomfort <- dat$discomfort

Xk <-model.matrix(tip~discomfort,dat)

yk <- as.matrix(dat[,1])

## time model

#dat <- data.frame(tip\_ratio = taxi$tip\_ratio,trip\_time\_in\_secs = taxi$trip\_time\_in\_secs)

#tip <- dat$tip\_ratio

#time <- dat$trip\_time\_in\_secs

#Xk <-model.matrix(tip~time,dat)

#yk <- as.matrix(dat[,1])

## distance model

#dat <- data.frame(tip\_ratio = taxi$tip\_ratio, trip\_distance=taxi$trip\_distance)

#tip <- dat$tip\_ratio

#distance <- dat$trip\_distance

#Xk <-model.matrix(tip~distance,dat)

#yk <- as.matrix(dat[,1])

### 2

## discomfort + time model

#dat <- data.frame(tip\_ratio = taxi$tip\_ratio,discomfort = taxi$discomfort,trip\_time\_in\_secs = taxi$trip\_time\_in\_secs)

#tip <- dat$tip\_ratio

#discomfort <- dat$discomfort

#time <- dat$trip\_time\_in\_secs

#Xk <-model.matrix(tip~discomfort+time,dat)

#yk <- as.matrix(dat[,1])

## discomfort + distance model

#dat <- data.frame(tip\_ratio = taxi$tip\_ratio,discomfort = taxi$discomfort,trip\_time\_in\_secs = taxi$trip\_time\_in\_secs, trip\_distance=taxi$trip\_distance)

#tip <- dat$tip\_ratio

#discomfort <- dat$discomfort

#distance <- dat$trip\_distance

#Xk <-model.matrix(tip~discomfort+distance,dat)

#yk <- as.matrix(dat[,1])

## time + distance model

#dat <- data.frame(tip\_ratio = taxi$tip\_ratio,discomfort = taxi$discomfort,trip\_time\_in\_secs = taxi$trip\_time\_in\_secs)

#tip <- dat$tip\_ratio

#time <- dat$trip\_time\_in\_secs

#distance <- dat$trip\_distance

#Xk <-model.matrix(tip~time+distance,dat)

#yk <- as.matrix(dat[,1])

### 3

## discomfort + time + distance model

#dat <- data.frame(tip\_ratio = taxi$tip\_ratio,discomfort = taxi$discomfort,trip\_time\_in\_secs = taxi$trip\_time\_in\_secs, trip\_distance=taxi$trip\_distance)

#tip <- dat$tip\_ratio

#discomfort <- dat$discomfort

#time <- dat$trip\_time\_in\_secs

#distance <- dat$trip\_distance

#Xk <-model.matrix(tip~discomfort+time+distance,dat)

#yk <- as.matrix(dat[,1])

###########################

XtXk <-crossprod(Xk,Xk)

Xtyk <-crossprod(Xk,yk)

ytyk <-crossprod(yk,yk)

res <- list(XtXk=XtXk,Xtyk=Xtyk,ytyk=ytyk)

keyval(1,res)

}

## 회귀분석 reduce 함수

reg.reduce.fun <- function(k,v){

XtX <- Reduce("+",v[seq\_along(v)%%3==1])

Xty <- Reduce("+",v[seq\_along(v)%%3==2])

yty <- Reduce("+",v[seq\_along(v)%%3==0])

res<-list(XtX=XtX,Xty=Xty,yty=yty)

keyval(1,res)

}

reg.mr <- mapreduce( input = files,

input.format = taxi.format,

map = reg.map.fun,

reduce = reg.reduce.fun)

dat <- values(from.dfs(m))

summary.fun <- function(v) {

XtX = v$XtX

Xty = v$Xty

yty = v$yty

nn = XtX[1,1]

beta.hat = solve(XtX, Xty)

ysum = Xty[1]

ybar = ysum/nn

stat <- list(nn = nn, beta.hat = beta.hat, ysum = ysum, ybar = ybar)

SST = yty - ysum^2/nn

SSE = yty - crossprod(beta.hat, Xty)

SSR = SST - SSE

SS <- list(SST = SST, SSE = SSE, SSR = SSR)

df.reg = dim(XtX)[1L] - 1

df.tot = nn - 1

df.res = df.tot - df.reg

DF <- list(df.reg = df.reg, df.tot = df.tot, df.res = df.res)

MSR = SSR / df.reg

MST = SST / df.tot

MSE = SSE / df.res

MS <- list(MSR = MSR, MST = MST, MSE = MSE)

f.val = MS$MSR / MS$MSE

p.val = pf(f.val, DF$df.reg, DF$df.res, lower.tail = F)

anova <- list(DF = DF, SS = SS, MS = MS, f.val = f.val, p.val = p.val)

res <- list(mat = v, stat = stat, anova = anova)

}

result <- summary.fun(dat)

result