CLC: Ctrl+L

Tobit:

# read response

vc<-read.csv("d:/firefoxproject/firefoxregressor.csv",header=F) # viewed category: 365 per day moving average last 10

d<-read.csv("d:/firefoxproject/Dwnld.csv",header=F) # downloads count (dayl one year): 365 days

dc<-read.csv("d:/firefoxproject/DocCategory.csv",header=F) #each review category frequency: 1212 items

s<-read.csv("d:/firefoxproject/Stars.csv",header=F) # stars for ordinal probit

s1<-read.csv("d:/firefoxproject/Stars1.csv",header=F) # one star (23)

s2<-read.csv("d:/firefoxproject/Stars2.csv",header=F) # two star (14)

s3<-read.csv("d:/firefoxproject/Stars3.csv",header=F) # Three star (56)

s4<-read.csv("d:/firefoxproject/Stars4.csv",header=F) # Four star (235)

s5<-read.csv("d:/firefoxproject/Stars5.csv",header=F) # Five star (884)

vc=as.matrix(vc)

d=as.matrix(d)

dc=as.matrix(dc)

s=as.matrix(s)

s1=as.matrix(s1)

s2=as.matrix(s2)

s3=as.matrix(s3)

s4=as.matrix(s4)

s5=as.matrix(s5)

#First models (frequentist)

#Gausian OLS

GOLSfit<-lm(d ~vc)

summary(GOLSfit)

#poisson regression

Poisfit <- glm(d ~ vc, family=poisson())

summary(Poisfit)

#First Models Bayesian

vcb<-cbind(1,vc)

dt1 <- list(y=d,X=vcb) #data for bayesian analysis

betabar1 <- as.numeric(coefficients(GOLSfit)) # c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) #priors

A1 <- 10 \* diag(21) # Pecision matrix for the normal prior. Again we have 2

n1 <- 21 # degrees of freedom for the inverse chi-square prior

ssq1 <- var(d) # scale parameter for the inverse chi-square prior

Prior1 <- list(betabar=betabar1, A=A1, nu=n1, ssq=ssq1)

iter <- 10000 # number of iterations of the Gibbs sampler

slice <- 1 # thinning/slicing parameter. 1 means we keep all all values

MCMC <- list(R=iter, keep=slice)

sim1 <- runiregGibbs(dt1, Prior1, MCMC)

pdf('BGOLSM.pdf',width=11, height=8.5,pointsize=12, paper='special') #pring multiple page of graph into one file

plot(sim1$betadraw)

dev.off()

pdf('BGOLSS.pdf',width=11, height=8.5,pointsize=12, paper='special') #pring multiple page of graph into one file

plot(sim1$sigmasqdraw)

dev.off()

summary(sim1$betadraw)

summary(sim1$sigmasqdraw)

#binary probit (i number of starst at specific level)

fit<-glm( stari~x, family=binomial(link=probit))

summary(fit)

#binary logit

fit<-glm(stari~x,family=binomial())

summary(fit)

#models (Bayesian)

dc<-cbind(1,dc)

#Gausian linear regression

#Tobit

#ordinary probit

#Binary probit

#binary Logit

library(survival)

example(tobin)

summary(tfit)

tfit.mcmc <- MCMCtobit(durable ~ age + quant, data=tobin, mcmc=30000,

verbose=1000)

plot(tfit.mcmc)

raftery.diag(tfit.mcmc)

summary(tfit.mcmc)

Logit:

n=1000;

k=4;

rn<-runif(n\*(k-1), min=0, max=1);

x<-matrix(rn,k-1,n);

xint=rbind(rep(1,n),x); # alternative: cbind(1,cov)

beta=c(1,-1,2);

s2=1;

y=rep(0,n);

zt=rep(0,n);

xint=t(xint);

x=t(x);

for (i in 1:n){

u=runif(1,min=0,max=1);

ei=sqrt(s2)\*log(u/(1-u));

zti=xint %\*% beta + ei;

zt[i]=zti;

if (zti>0) y[i]=1;

}

library(mcmc)

out<-glm(y~x,family=binomial())

summary(out);

# intrcpt= 2.98 x1=NA x2=-0.2241 x3=-0.5037

x<-cbind(1,x);

lupost<-function(beta,x,y){

eta <- x%\*%beta

p<-1/(1+exp(-eta))

logl<-sum(log(p[y==1]))+sum(log(1-p[y==0]))

return(logl+sum(dnorm(beta,0,2,log=TRUE)))

}

set.seed(42)

beta.init <- as.numeric(coefficients(out))

out <- metrop(lupost, beta.init, 1000, x=x, y=y)

names(out)

out$accept

out <-metrop(out, scale=0.1, x=x,y=y);

out$accept

out <- metrop(out, scale = 0.3, x = x, y = y)

out$accept

out <- metrop(out, scale = 0.5, x = x, y = y)

out$accept

out <- metrop(out, scale = 0.4, x = x, y = y)

out$accept

out <- metrop(out, nbatch = 10000, x = x, y = y)

out$accept

out$time

plot(ts(out$batch))

acf(out$batch)

# library code

library(mcmc)

data(logit)

out<-glm(y~x1+x2+x3+x4,data=logit, family=binomial());

summary(out);

x<-logit;

x$y<-NULL;

x<-as.matrix(x);

x<-cbind(1,x);

dimnames(x)<-NULL;

y<-logit$y;

lupost<-function(beta,x,y){

eta <- x%\*%beta

p<-1/(1+exp(-eta))

logl<-sum(log(p[y==1]))+sum(log(1-p[y==0]))

return(logl+sum(dnorm(beta,0,2,log=TRUE)))

}

set.seed(42)

beta.init <- as.numeric(coefficients(out))

out <- metrop(lupost, beta.init, 1000, x=x, y=y)

names(out)

out$accept

#clear workspace

rm(list = ls());

rm(list = ls()[grep("^tmp", ls())])

rm(list=ls(pattern="^tmp"))

rm(list = grep("^paper", ls(), value = TRUE, invert = TRUE))

Pasted from <<http://stackoverflow.com/questions/11761992/remove-data-from-workspace>>

Statistical inference Homework:

retailSales<-read.csv("d:/HW12TS.csv",header=T) # read whole data

# from Jan 1955 to Dec 1975 as a time series object

rsts <- ts(retailSales, start=c(1955, 1), end=c(1977, 12), frequency=12)

#excluding last two years

rtsm <- window(rsts, start=c(1955,1), end=c(1975,12))

#plot time series

plot(rsts)

plot.ts(rtsm)

#convert to additive

lrtsm <- log(rtsm)

plot.ts(lrtsm)

# Decomposition

library("TTR")

lrtsm6 <- SMA(lrtsm,n=6)

plot.ts(lrtsm6)

lrtsm12 <- SMA(lrtsm,n=12)

plot.ts(lrtsm12)

#Decomposition

lrtsmcomp<- decompose(lrtsm)

# Holt winter exponential smoothing

lrtsmforecast<- HoltWinters(lrtsm, beta=FALSE, gamma=FALSE)

lrtsmforecast$SSE

# forecast without data

library("forecast")

hwfwd <- forecast.HoltWinters(lrtsmforecast, h=24)

plot(hwfwd)

accuracy(hwfwd ) # predictive accuracy

#plot predicted value versus real value

rtsmf <- window(rsts, start=c(1976,1), end=c(1977,12))

rtsmfl<-log(rtsmf)

confl95<-hwfwd$lower[,2]

plot(confl95, type="l" , col=2)

par(new=T)

confu95<-hwfwd$upper[,2]

plot(confu95, type="l" , col=2)

par(new=T)

plot(rtsmfl,type="b",axes=F,col=3)

par(new=F)

#check correlogram

acf(hwfwd$residuals, lag.max=20)

#Ljung-Box test

Box.test(hwfwd$residuals, lag=20, type="Ljung-Box")

plot.ts(hwfwd$residuals)

#Check normality of residuals

plotForecastErrors <- function(forecasterrors)

{

# make a histogram of the forecast errors:

mybinsize <- IQR(forecasterrors)/4

mysd <- sd(forecasterrors)

mymin <- min(forecasterrors) - mysd\*5

mymax <- max(forecasterrors) + mysd\*3

# generate normally distributed data with mean 0 and standard deviation mysd

mynorm <- rnorm(10000, mean=0, sd=mysd)

mymin2 <- min(mynorm)

mymax2 <- max(mynorm)

if (mymin2 < mymin) { mymin <- mymin2 }

if (mymax2 > mymax) { mymax <- mymax2 }

# make a red histogram of the forecast errors, with the normally distributed data overlaid:

mybins <- seq(mymin, mymax, mybinsize)

hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)

# freq=FALSE ensures the area under the histogram = 1

# generate normally distributed data with mean 0 and standard deviation mysd

myhist <- hist(mynorm, plot=FALSE, breaks=mybins)

# plot the normal curve as a blue line on top of the histogram of forecast errors:

points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)

}

plotForecastErrors(hwfwd$residuals)

qqnorm(hwfwd$residuals, ylab="Holt Winter Multiplicative", xlab="Normal Scores", main="Notmsl pp Holt Winter Multiplicative")

qqline(hwfwd$residuals)

# ARIMA

lrtsm1 <- diff(lrtsm , differences=1)

plot.ts(lrtsm1)

lrtsm2 <- diff(lrtsm , differences=2)

plot.ts(lrtsm2)

# determine degree of ARIMA

acf(lrtsm1 , lag.max=20) #11

pacf(lrtsm1, lag.max=20) #19

# fit an ARIMA model of order P, D, Q

fit <- arima(lrtsm, order=c(0, 1, 11))

accuracy(fit) # predictive accuracy

forecast(fit, 24)

plot(forecast(fit, 24))

# Automated forecasting using an ARIMA model

fit <- auto.arima(lrtsm)

accuracy(fit) # predictive accuracy

forecast(fit, 24)

plot(forecast(fit, 24))

# Automated forecasting using an exponential model

fit <- ets(lrtsm)

accuracy(fit) # predictive accuracy

forecast(fit, 24)

plot(forecast(fit, 24))

spectrum(rsts)

Statistical Inference, Matrix:

crime<-read.csv("d:/eleventh.csv",header=T) # read whole data

cr=crime[,1] # crime rate

cov=crime[,2:14] # covariates

summary(cov)# check summary of data

cor(cov) #check the correlation

pairs(cov) # check plot of correlations

plot(cov[,1], cr, main="Age and crime rate", xlab="Age", ylab="Crime rate", pch=19)

plot(cov[,2], cr, main="Southern states and crime rate", xlab="Southern states", ylab="Crime rate", pch=19)

plot(cov[,3], cr, main="Schooling and crime rate", xlab="Schooling", ylab="Crime rate", pch=19)

plot(cov[,4], cr, main="1960 plice expenditure", xlab="1960 police expenditure", ylab="Crime rate", pch=19)

plot(cov[,5], cr, main="1959 plice expenditure", xlab="1959 police expenditure", ylab="Crime rate", pch=19)

plot(cov[,6], cr, main="Labor force participation", xlab="Labor force participation", ylab="Crime rate", pch=19)

plot(cov[,7], cr, main="Male/Female rate", xlab="male/female rate", ylab="Crime rate", pch=19)

plot(cov[,8], cr, main="State population", xlab="State population", ylab="Crime rate", pch=19)

plot(cov[,9], cr, main="Non white", xlab="non white", ylab="Crime rate", pch=19)

plot(cov[,10], cr, main="Unemployment rate 14-24", xlab="unemployment rate 14-24", ylab="Crime rate", pch=19)

plot(cov[,11], cr, main="Unemployment rate 25-35", xlab="unemployment rate 25-35", ylab="Crime rate", pch=19)

plot(cov[,12], cr, main="Transferable goods and assets", xlab="Transferable goods and assets", ylab="Crime rate", pch=19)

plot(cov[,13], cr, main="Below median income", xlab="Families Below median income", ylab="Crime rate", pch=19)

qqnorm(cr, ylab="Crime rate", xlab="Normal Scores", main="Normal probability plot of crime rate")

qqline(cr)

names(cov)

crd = d=as.matrix(cr)

covd = d=as.matrix(cov)

cdata=cbind(cr,cov)

model1 = lm(cr~Age+S+Ed+Ex0+Ex1+LF+M+N+NW+U1+U2+W+X,data=cdata)

step(model1, direction="backward")

step(model1, direction="forward")

step(model1, direction="both",trace=TRUE)

par(mfrow=c(2,2)) # visualize four graphs at once

plot(model1)

par(mfrow=c(1,1)) # reset the graphics defaults

Summary Stat: Summer Project

# first row contains variable names

# we will read in workSheet mysheet

library(RODBC)

channel <- odbcConnectExcel("C:/Users/MHE/Desktop/ActiveCourses/Projects/Noris/Data/DailyOf100AddOn.xlsx")

mydata <- sqlFetch(channel, "CrossSec")

dwnldDTA<-sqlFetch(channel, "Summary")

odbcClose(channel)

cbind(summary(mydata))

names(mydata)

newdata <- mydata[c(0,5:23)]

summary(newdata[c(0,19)])

#other method to get summery statistics

library(Hmisc)

describe(mydata)

#btter method to get summery

library(pastecs)

stat.desc(mydata)

#the best way to use the summery (like SAS)

library(psych)

describe(mydata)

describe(dwnldDTA)

cbind(table(newdata[c(0,10)]))

summary(dwnldDTA[c(0,11)])

library(MASS) # load the MASS package

cbind(table(mydata$"2nd Category")) # apply the table functionc

Summary Stat: Summer Project

mydata = read.csv("D:/FirFxPrl/sampletest.csv")

library("TTR")

Downloads= log(EMA(mydata[2], 7));

St\_AVG= EMA(mydata[16], 7);

St\_Cnt= EMA(mydata[17], 7);

St\_STD= EMA(mydata[18], 7);

Usage= log(EMA(mydata[9], 7));

English.Share= EMA(mydata[15], 7);

model1=lm(Downloads~St\_AVG+St\_Cnt+St\_STD+Usage+English.Share)

summary(model1)

st=as.data.frame(mydata)

str(st)

cor(st)

pairs(st)

model1=lm(log(Downloads)~St\_AVG+St\_Cnt+St\_STD+log(Usage)+English.Share,data=st)

summary(model1)

MCMC part of code of Bayesian BLP:

# ------------ (1) Gibbs Sampler for thetabar and taosq -------------------output=runiregG(y=mu,X=X,XpX=XpX,Xpy=crossprod(X,mu),sigmasq=taosq,

A=Athetabar,betabar=thetabar0,nu=nu0,ssq=s0sq) thetabar=output$betadraw

taosq=output$sigmasqdraw

# ------------ (2) Metropolis for r ---------------------------------------#

Random-Walk Chain

rN=r+mvrnorm(1,rep(0,(K\*(K+1)/2)),varn

\_r)\_

ON=Loglhd(rN,mu,thetabar,taosq)

prior\_old=sum(-r[1:K]^2/2/sigmasqR\_DIAG)+sum(-r[(K+1):(K\*(K+1)/2)]^2/2/sigmasqR\_off)

prior\_new=sum(-rN[1:K]^2/2/sigmasqR\_DIAG)+sum(-rN[(K+1):(K\*(K+1)/2)]^2/2/sigmasqR\_off)

15

# Evaluate old r (mu) at new (thetabar,taosq)

eta=mu-X%\*%thetabar

llhd\_old=sum(log(dnorm(eta,sd=sqrt(taosq))))+OO$sumlogjacob

ratio=exp(ON$llhd+prior\_new-llhd\_old-prior\_old)

alphaS=min(1,ratio) # S stands for Sigma

if (runif(1)<=alphaS) {

r=rN; OO=ON; ns=ns+1; mu=OO$mu

r=rN; OO=ON; ns=ns+1; mu=OO$mu

}

Brute-Force log-likelihood code of Bayesian BLP:

# Purpose: Evaluate log likelihood. Sigma is reparameterized as r.

Loglhd slow = function(thetabar,r,taosq,mu){

# (1). Transform r to L, where Sigma=LL'

L=diag(exp(r[1:K]))

L[lower tri(L)]=r[(K+1):(K\*(K+1)/2)]L[lower.tri(L)]=r[(K+1):(K (K+1)/2)]

# (2). At given L, do inversion to get mu. Then compute eta

temp=invert\_slow(L,mu,v,crit,T,H,J,lnactS,indTHJ,indJTH)

mu = temp$mu; prob = temp$prob; niter = temp$niter

eta=mu-X%\*%thetabar

eta=mu-X%\*%thetabar

# (3). Jacobian

# Form J diagonal elements at each time t

diagonal=rowMeans(prob\*(1-prob)) # TJ by 1 vector

# Form the off diagonal elements

dd=-prob%\*%t(prob)/H # TJ by TJ

cc=aaa\*dd+diag(diagonal)#TJ by TJ matrix: block diagonal

for (t in 1:T){

for (t in 1:T){

cct=cc[((t-1)\*J+1):(t\*J),((t-1)\*J+1):(t\*J)] #(t)th block of cc

logjacob[t]=-log(abs(det(cct)))

}

# (4). Form Log Likelihood

lj b (lj b)

sumlogjacob=sum(logjacob)

llhd=sum(log(dnorm(eta,sd=sqrt(taosq))))+sumlogjacob

list(llhd=llhd,mu=mu,niter=niter,sumlogjacob=sumlogjacob)

}

Slow inversion code of Bayesian BLP

invert\_slow =

function(L,mu,v,crit,T,H,J,lnactS,indTHJ,indJTH){

# Purpose: Invert observed shares S at give L to get mean utility mu's.

niter=0 # number of iterations taken for the inversion

munew=mu # starting value

muold=munew/2

upart=X%\*%L%\*%v

hil ( (b(( ld )/ ))> it){

while (max(abs((muold-munew)/munew))>crit){

muold=munew

num=exp(upart+ muold) # JT by H numerator

den1=matrix(double(T\*H),nrow=T)

for (t in 1:T){

den1[t,]=1+colSums(num[((t-1)\*J+1):(t\*J),]) #T by H

}

den=matrix(rep(den1,each=J),ncol=H) #replc each t J times,JT by H

prob=num/den # JT by H

sh=t(matrix(rowMeans(prob), nrow=J)) # T by J predicted share

munew=t(matrix(muold,nrow=J))+log(S)-log(sh) # T by J

munew=as.vector(t(munew)) # length JT vector

niter=niter+1

}

List(mu=munew,prob=prob,niter=niter)

}

Add new row to the object res each time it finds a row that has no NA

funAgg = function(x) {  
# initialize res   
 res = NULL  
 n = nrow(x)

for (i in 1:n) {  
 if (!any(is.na(x[i,]))) res = rbind(res, x[i,])  
 }  
 res  
}

Pasted from <<http://www.stat.berkeley.edu/~nolan/stat133/Fall05/lectures/profilingEx.html>>

Initialize res object to the correct size, replace the rwos one at a time as a row in the nput with no Nas is found

funLoop = function(x) {  
# Initialize res with x  
 res = x  
 n = nrow(x)  
 k = 1

for (i in 1:n) {  
 if (!any(is.na(x[i,]))) {  
 res[k, ] = x[i,]  
 k = k + 1  
 }  
 }  
 res[1:(k-1), ]  
}

Pasted from <<http://www.stat.berkeley.edu/~nolan/stat133/Fall05/lectures/profilingEx.html>>

Is.na function that returns a logical when given a data frame of data

funApply = function(x) {  
 drop = apply(is.na(x), 1, any)  
 x[!drop, ]  
}

Pasted from <<http://www.stat.berkeley.edu/~nolan/stat133/Fall05/lectures/profilingEx.html>>

Profiling:

funOmit = function(x) {  
# The or operation is very fast, it is replacing the any function  
# Also note that it doesn't require having another data frame as big as x

drop = F  
 n = ncol(x)  
 for (i in 1:n)  
 drop = drop | is.na(x[, i])  
 x[!drop, ]  
}

#Make up large test case  
 xx = matrix(rnorm(2000000),100000,20)  
 xx[xx>2] = NA  
 x = as.data.frame(xx)

# Call the R code profiler and give it an output file to hold results  
 Rprof("exampleAgg.out")  
# Call the function to be profiled  
 y = funAgg(xx)  
 Rprof(NULL)

Rprof("exampleLoop.out")  
 y = funLoop(xx)  
 Rprof(NULL)

Rprof("exampleApply.out")  
 y = funApply(xx)  
 Rprof(NULL)

Rprof("exampleOmit.out")  
 y = funOmit(xx)  
 Rprof(NULL)

Pasted from <<http://www.stat.berkeley.edu/~nolan/stat133/Fall05/lectures/profilingEx.html>>

Vectorized invert: faster for BLP Bayesian

invert =invert

function(L,mu,v,crit,T,H,J,lnactS,indTHJ,indJTH){

# Purpose: Invert observed shares S at give L to get mean utility mu's.

19

niter=0 # number of iterations taken

munew=mu # starting value

muold=munew/2

upart=X%\*%L%\*%v

while (max(abs((muold-munew)/munew))>crit){

muold=munew

num=exp(upart+ muold) # num is JT x H

dim(num)=NULL # convert num to JTH vector

nmnm[indTHJ] # con ert n m to THJ ectornum=num[indTHJ] # convert num to THJ vector

dim(num)=c(T\*H,J) # convert num to TH \* J matrix

den=1+rowSums(num) # TH vector

prob=num/den # TH \* J matrix

dim(prob)=NULL # convert prob to THJ vector

prob=prob[indJTH] # convert prob to JTH vector

dim(prob)=c(J\*T,H) # convert prob to JT \* H matrix

sh=rowMeans(prob) # JT vector

munew=muold+lnactS-log(sh)# JT vector

niter=niter+1

}

niter=niter+1

list(mu=munew,prob=prob,niter=niter)

}

No loop!

Matrix divided

by a vector

Global variable inside the function:

a <- "old"  
test <- function () {  
 assign("a", "new", envir = .GlobalEnv)  
}  
test()  
a # display the new value

Pasted from <<http://stackoverflow.com/questions/1236620/global-variables-in-r>>

a <<- "new"

Pasted from <<http://stackoverflow.com/questions/1236620/global-variables-in-r>>

Reindiexting Function BLP Bayesian:

JTH THJ=function(J H T){JTH\_THJ=function(J,H,T){

#

# function to convert and index a vector ordered j by t by h (i.e. j

# varies faster than t than h) into a vector ordered t by h by j

#

ind=double(J\*H\*T)

cnt=1

for (j in 1:J){

for (h in 1:H) {

for (t in 1:T) {for (t in 1:T) {

ind[cnt]=(t-1)\*J+(h-1)\*(T\*J)+j

cnt=cnt+1

}

}

}

return(ind)

}

indTHJ JTH\_THJ(J,H,T) indJTH=THJ\_JTH(J,H,T)

indTHJ=JTH THJ(J,H,T)

New Code:

for (t in 1:T){

cct=cc[((t 1)\*J+1):(t\*J) ((t 1)\*J+1):(t\*J)] #(t)th block of cccct=cc[((t-1)\*J+1):(t\*J),((t-1)\*J+1):(t\*J)] #(t)th block of cc

logjacob[t]=-2\*sum(log(diag(chol(cct))))

# old code:

# logjacob[t]=-log(abs(det(cct)))

}

Creating Matrix in R:

**seq1 <- seq(1:6)  
mat1 <- matrix(seq1, 2)  
mat1**  
 [,1] [,2] [,3]   
[1,] 1 3 5  
[2,] 2 4 6

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

#filling the matrix by rows  
**mat2 <- matrix(seq1, 2, byrow = T)  
mat2**  
 [,1] [,2] [,3]   
[1,] 1 2 3  
[2,] 4 5 6

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

# Number of columns without specifying rows

**mat3 <- matrix(seq1, ncol = 2)  
mat3**

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

#creating the same matrix using both dimension arguments  
#by using them in order we do not have to name them  
**mat4 <- matrix(seq1, 3, 2)  
mat4**  
 [,1] [,2]   
[1,] 1 4  
[2,] 2 5  
[3,] 3 6

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

#creating a matrix of 20 numbers from a standard normal dist.  
**mat5 <- matrix(rnorm(20), 4)  
mat5**

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

#appending v1 to mat5  
**v1 <- c(1, 1, 2, 2)  
mat6 <- cbind(mat5, v1)  
mat6**

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

**v2 <- c(1:6)  
mat7 <- rbind(mat6, v2)  
mat7**

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

#determining the dimensions of a mat7  
**dim(mat7)**  
[1] 5 6

#removing names of rows and columns  
#the first NULL refers to all row names, the second to all column names   
**dimnames(mat7) <- list(NULL, NULL)  
mat7**

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

Access element of a matrix:

matrix\_name[row#, col#]

**mat7[1, 6]**  
[1] 1

#to access an entire row leave the column number blank  
**mat7[1, ]**  
[1] -0.1920780 0.0910308 -1.1044547 -1.1513583 1.3435247 1.0000000

#to access an entire column leave the row number blank  
**mat7[, 6]**  
[1] 1 1 2 2 6

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

#Creating mat8 and mat9  
**mat8 <- matrix(1:6, 2)  
mat8**  
 [,1] [,2] [,3]   
[1,] 1 3 5  
[2,] 2 4 6

**mat9 <- matrix(c(rep(1, 3), rep(2, 3)), 2, byrow = T)  
mat9**  
 [,1] [,2] [,3]   
[1,] 1 1 1  
[2,] 2 2 2

#addition  
**mat9 + mat8**  
 [,1] [,2] [,3]   
[1,] 2 4 6  
[2,] 4 6 8

**mat9 + 3**  
 [,1] [,2] [,3]   
[1,] 4 4 4  
[2,] 5 5 5

#subtraction  
**mat8 - mat9**  
 [,1] [,2] [,3]   
[1,] 0 2 4  
[2,] 0 2 4

#inverse  
**solve(mat8[, 2:3])**  
 [,1] [,2]   
[1,] -3 2.5  
[2,] 2 -1.5

#transpose  
**t(mat9)**  
 [,1] [,2]   
[1,] 1 2  
[2,] 1 2  
[3,] 1 2

#multiplication  
#we transpose mat8 since the dimension of the matrices have to match  
#dim(2, 3) times dim(3, 2)  
**mat8 %\*% t(mat9)**  
 [,1] [,2]   
[1,] 9 18  
[2,] 12 24

#element-wise multiplication  
**mat8 \* mat9**  
 [,1] [,2] [,3]   
[1,] 1 3 5  
[2,] 4 8 12

**mat8 \* 4**  
 [,1] [,2] [,3]   
[1,] 4 12 20  
[2,] 8 16 24

#division  
**mat8/mat9**  
 [,1] [,2] [,3]   
[1,] 1 3 5  
[2,] 1 2 3

**mat9/2**  
 [,1] [,2] [,3]   
[1,] 0.5 0.5 0.5  
[2,] 1.0 1.0 1.0

#using submatrices from the same matrix in computations  
**mat8[, 1:2]**  
 [,1] [,2]   
[1,] 1 3  
[2,] 2 4

**mat8[, 2:3]**  
 [,1] [,2]   
[1,] 3 5  
[2,] 4 6

**mat8[, 1:2]/mat8[, 2:3]**  
 [,1] [,2]   
[1,] 0.3333333 0.6000000  
[2,] 0.5000000 0.6666667

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

Regression:

**y <- matrix(hsb2$write, ncol = 1)  
y[1:10, 1]**  
 [1] 52 59 33 44 52 52 59 46 57 55

**x <- as.matrix(cbind(1, hsb2$math, hsb2$science, hsb2$socst, hsb2$female))  
x[1:10, ]**  
 [,1] [,2] [,3] [,4] [,5]   
 [1,] 1 41 47 57 0  
 [2,] 1 53 63 61 1  
 [3,] 1 54 58 31 0  
 [4,] 1 47 53 56 0  
 [5,] 1 57 53 61 0  
 [6,] 1 51 63 61 0  
 [7,] 1 42 53 61 0  
 [8,] 1 45 39 36 0  
 [9,] 1 54 58 51 0  
[10,] 1 52 50 51 0

**n <- nrow(x)  
p <- ncol(x)**

#parameter estimates  
**beta.hat <- solve(t(x) %\*% x) %\*% t(x) %\*% y  
beta.hat**  
 [,1]   
[1,] 6.5689235  
[2,] 0.2801611  
[3,] 0.2786543  
[4,] 0.2681117  
[5,] 5.4282152

#predicted values  
**y.hat <- x %\*% beta.hat  
y.hat[1:5, 1]**  
[1] 46.43465 60.75571 46.17103 49.51943 53.66160

**y[1:5, 1]**  
[1] 52 59 33 44 52

#the variance, residual standard error and df's  
**sigma2 <- sum((y - y.hat)^2)/(n - p)**

#residual standard error  
**sqrt(sigma2)**  
[1] 6.101191

#degrees of freedom  
**n - p**  
[1] 195

#the standard errors, t-values and p-values for estimates  
#variance/covariance matrix  
**v <- solve(t(x) %\*% x) \* sigma2**

#standard errors of the parameter estimates  
**sqrt(diag(v))**  
[1] 2.81907949 0.06393076 0.05804522 0.04919499 0.88088532

#t-values for the t-tests of the parameter estimates  
**t.values <- beta.hat/sqrt(diag(v))  
t.values**  
 [,1]   
[1,] 2.330166  
[2,] 4.382257  
[3,] 4.800642  
[4,] 5.449980  
[5,] 6.162227

#p-values for the t-tests of the parameter estimates  
**2 \* (1 - pt(abs(t.values), n - p))**  
[1] 2.082029e-002 1.917191e-005 3.142297e-006 1.510015e-007 4.033511e-009

#checking that we got the correct results  
**ex1 <- lm(write ~ math + science + socst + female, hsb2)  
summary(ex1)**

Call: lm(formula = write ~ math + science + socst + female, data = hsb2)

Coefficients:  
 Value Std. Error t value Pr(>|t|)   
(Intercept) 6.5689 2.8191 2.3302 0.0208   
 math 0.2802 0.0639 4.3823 0.0000   
 science 0.2787 0.0580 4.8006 0.0000   
 socst 0.2681 0.0492 5.4500 0.0000   
 female 5.4282 0.8809 6.1622 0.0000

Residual standard error: 6.101 on 195 degrees of freedom  
Multiple R-Squared: 0.594   
F-statistic: 71.32 on 4 and 195 degrees of freedom, the p-value is 0

Pasted from <<http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm>>

Single Dimension of a Matrix:

# simple versions of nrow and ncol could be defined as follows  
nrow0 <- function(x) dim(x)[1]  
ncol0 <- function(x) dim(x)[2]

Pasted from <<http://stat.ethz.ch/R-manual/R-devel/library/base/html/dim.html>>

Vector of zeros or single value:

list(rep(0, 50))

Pasted from <<http://stackoverflow.com/questions/4072388/how-do-you-create-a-list-with-a-single-value-in-r>>

as.list(rep(0, 50))

Pasted from <<http://stackoverflow.com/questions/4072388/how-do-you-create-a-list-with-a-single-value-in-r>>

Display a variable with its name:

names(mydata)[3] <- "This is the label for variable 3"

mydata[3] # list the variable

library(Hmisc)

label(mydata$myvar) <- "Variable label for variable *myvar*"

describe(mydata)

Pasted from <<http://www.statmethods.net/input/variablelables.html>>

Pasted from <<http://www.statmethods.net/input/variablelables.html>>

Input and Display:

#read files with labels in first row  
read.table(filename,header=TRUE) #read a tab or space delimited file  
read.table(filename,header=TRUE,sep=',') #read csv files

x=c(1,2,4,8,16 ) #create a data vector with specified elements  
y=c(1:10) #create a data vector with elements 1-10  
n=10  
x1=c(rnorm(n)) #create a n item vector of random normal deviates  
y1=c(runif(n))+n #create another n item vector that has n added to each random uniform distribution  
z=rbinom(n,size,prob) #create n samples of size "size" with probability prob from the binomial  
vect=c(x,y) #combine them into one vector of length 2n  
mat=cbind(x,y) #combine them into a n x 2 matrix  
mat[4,2] #display the 4th row and the 2nd column  
mat[3,] #display the 3rd row  
mat[,2] #display the 2nd column  
subset(dataset,logical) #those objects meeting a logical criterion  
subset(data.df,select=variables,logical) #get those objects from a data frame that meet a criterion  
data.df[data.df=logical] #yet another way to get a subset  
x[order(x$B),] #sort a dataframe by the order of the elements in B  
x[rev(order(x$B)),] #sort the dataframe in reverse order

browse.workspace #a menu command that creates a window with information about all variables in the workspace

Pasted from <<https://personality-project.org/r/r.commands.html>>

Moving Around in the workspace:

ls() #list the variables in the workspace  
rm(x) #remove x from the workspace  
rm(list=ls()) #remove all the variables from the workspace  
attach(mat) #make the names of the variables in the matrix or data frame available in the workspace  
detach(mat) #releases the names  
new=old[,-n] #drop the nth column  
new=old[n,] #drop the nth row  
new=subset(old,logical) #select those cases that meet the logical condition  
complete = subset(data.df,complete.cases(data.df)) #find those cases with no missing values  
new=old[n1:n2,n3:n4] #select the n1 through n2 rows of variables n3 through n4)

Pasted from <<https://personality-project.org/r/r.commands.html>>

Data Manipulation

replace(x, list, values) #remember to assign this to some object i.e., x <- replace(x,x==-9,NA)   
 #similar to the operation x[x==-9] <- NA

cut(x, breaks, labels = NULL,  
 include.lowest = FALSE, right = TRUE, dig.lab = 3, ...)

x.df=data.frame(x1,x2,x3 ...) #combine different kinds of data into a data frame  
        as.data.frame()  
        is.data.frame()  
x=as.matrix()  
scale() #converts a data frame to standardized scores

round(x,n) #rounds the values of x to n decimal places  
ceiling(x) #vector x of smallest integers > x  
floor(x) #vector x of largest interger < x  
as.integer(x) #truncates real x to integers (compare to round(x,0)  
as.integer(x < cutpoint) #vector x of 0 if less than cutpoint, 1 if greater than cutpoint)  
factor(ifelse(a < cutpoint, "Neg", "Pos")) #is another way to dichotomize and to make a factor for analysis   
transform(data.df,variable names = some operation) #can be part of a set up for a data set

x%in%y #tests each element of x for membership in y  
y%in%x #tests each element of y for membership in x  
all(x%in%y) #true if x is a proper subset of y  
all(x) # for a vector of logical values, are they all true?  
any(x) #for a vector of logical values, is at least one true?

Pasted from <<https://personality-project.org/r/r.commands.html>>

Distributions:

beta(a, b)  
gamma(x)  
choose(n, k)  
factorial(x)

dnorm(x, mean=0, sd=1, log = FALSE) #normal distribution  
pnorm(q, mean=0, sd=1, lower.tail = TRUE, log.p = FALSE)  
qnorm(p, mean=0, sd=1, lower.tail = TRUE, log.p = FALSE)  
rnorm(n, mean=0, sd=1)

dunif(x, min=0, max=1, log = FALSE) #uniform distribution  
punif(q, min=0, max=1, lower.tail = TRUE, log.p = FALSE)  
qunif(p, min=0, max=1, lower.tail = TRUE, log.p = FALSE)  
runif(n, min=0, max=1)

rnorm(n,mean,sd)  
rbinom(n,size,p)  
sample(x, size, replace = FALSE, prob = NULL) #samples with or without replacement

Pasted from <<https://personality-project.org/r/r.commands.html>>

Pasted from <<https://personality-project.org/r/r.commands.html>>

Statistics and Transformations:

max()  
min()  
mean()  
median()  
sum()  
var() #produces the variance covariance matrix  
sd() #standard deviation  
mad() #(median absolute deviation)  
fivenum() #Tukey fivenumbers min, lowerhinge, median, upper hinge, max  
table() #frequency counts of entries, ideally the entries are factors(although it works with integers or even reals)  
scale(data,scale=T) #centers around the mean and scales by the sd)  
cumsum(x) #cumulative sum, etc.  
cumprod(x)  
cummax(x)  
cummin(x)  
rev(x) #reverse the order of values in x  
   
cor(x,y,use="pair") #correlation matrix for pairwise complete data, use="complete" for complete cases  
   
aov(x~y,data=datafile) #where x and y can be matrices  
aov.ex1 = aov(DV~IV,data=data.ex1) #do the analysis of variance or  
aov.ex2 = aov(DV~IV1\*IV21,data=data.ex2) #do a two way analysis of variance  
summary(aov.ex1) #show the summary table  
print(model.tables(aov.ex1,"means"),digits=3) #report the means and the number of subjects/cell  
boxplot(DV~IV,data=data.ex1) #graphical summary appears in graphics window

lm(x~y,data=dataset) #basic linear model where x and y can be matrices (see plot.lm for plotting options)  
t.test(x,g)  
pairwise.t.test(x,g)  
power.anova.test(groups = NULL, n = NULL, between.var = NULL,  
 within.var = NULL, sig.level = 0.05, power = NULL)  
power.t.test(n = NULL, delta = NULL, sd = 1, sig.level = 0.05,  
 power = NULL, type = c("two.sample", "one.sample", "paired"),  
 alternative = c("two.sided", "one.sided"),strict = FALSE)

Pasted from <<https://personality-project.org/r/r.commands.html>>

Regression and Linear Models:

matrices  
lm(Y~X1+X2)  
lm(Y~X|W)   
solve(A,B) #inverse of A \* B - used for linear regression  
solve(A) #inverse of A  
factanal()  
princomp()

Pasted from <<https://personality-project.org/r/r.commands.html>>

Rowsum and columnsum and min and max:

colSums (x, na.rm = FALSE, dims = 1)  
 rowSums (x, na.rm = FALSE, dims = 1)  
 colMeans(x, na.rm = FALSE, dims = 1)  
 rowMeans(x, na.rm = FALSE, dims = 1)  
 rowsum(x, group, reorder = TRUE, ...) #finds row sums for each level of a grouping variable  
 apply(X, MARGIN, FUN, ...) #applies the function (FUN) to either rows (1) or columns (2) on object X  
         apply(x,1,min) #finds the minimum for each row  
         apply(x,2,max) #finds the maximum for each column  
 col.max(x) #another way to find which column has the maximum value for each row   
 which.min(x)  
 which.max(x)  
         z=apply(big5r,1,which.min) #tells the row with the minimum value for every column

Pasted from <<https://personality-project.org/r/r.commands.html>>

Working with Dates:

date <-strptime(as.character(date), "%m/%d/%y") #change the date field to a internal form for time   
 #see ?formats and ?POSIXlt   
 as.Date  
 month= months(date) #see also weekdays, Julian

Pasted from <<https://personality-project.org/r/r.commands.html>>

Graphics:

par(mfrow=c(nrow,mcol)) #number of rows and columns to graph  
par(ask=TRUE) #ask for user input before drawing a new graph  
par(omi=c(0,0,1,0) ) #set the size of the outer margins   
mtext("some global title",3,outer=TRUE,line=1,cex=1.5) #note that we seem to need to add the global title last  
 #cex = character expansion factor

boxplot(x,main="title") #boxplot (box and whiskers)

title( "some title") #add a title to the first graph

hist() #histogram  
plot()  
        plot(x,y,xlim=range(-1,1),ylim=range(-1,1),main=title)  
        par(mfrow=c(1,1)) #change the graph window back to one figure  
        symb=c(19,25,3,23)  
        colors=c("black","red","green","blue")  
        charact=c("S","T","N","H")  
        plot(PA,NAF,pch=symb[group],col=colors[group],bg=colors[condit],cex=1.5,main="Postive vs. Negative Affect by Film condition")  
        points(mPA,mNA,pch=symb[condit],cex=4.5,col=colors[condit],bg=colors[condit])  
          
curve()  
abline(a,b)  
         abline(a, b, untf = FALSE, ...)  
 abline(h=, untf = FALSE, ...)  
 abline(v=, untf = FALSE, ...)  
 abline(coef=, untf = FALSE, ...)  
 abline(reg=, untf = FALSE, ...)

identify()  
        plot(eatar,eanta,xlim=range(-1,1),ylim=range(-1,1),main=title)  
        identify(eatar,eanta,labels=labels(energysR[,1]) ) #dynamically puts names on the plots  
locate()

legend()  
pairs() #SPLOM (scatter plot Matrix)  
pairs.panels () #SPLOM on lower off diagonal, histograms on diagonal, correlations on diagonal  
 #not standard R, but uses a function found in useful.r   
matplot ()  
biplot ())  
plot(table(x)) #plot the frequencies of levels in x

x= recordPlot() #save the current plot device output in the object x  
replayPlot(x) #replot object x  
dev.control #various control functions for printing/saving graphic files  
pdf(height=6, width=6) #create a pdf file for output  
dev.of() #close the pdf file created with pdf   
layout(mat) #specify where multiple graphs go on the page  
 #experiment with the magic code from Paul Murrell to do fancy graphic location  
layout(rbind(c(1, 1, 2, 2, 3, 3),  
 c(0, 4, 4, 5, 5, 0)))   
for (i in 1:5) {  
 plot(i, type="n")  
 text(1, i, paste("Plot", i), cex=4)  
}

pie(rep(1,16),col=rainbow(16))

> z <- seq(-3,3,.1)  
> d <- dnorm(z)  
> plot(z,d,type="l")  
> title("The Standard Normal Density",col.main="cornflowerblue")

Pasted from <<http://data.princeton.edu/R/gettingStarted.html>>

Pasted from <<https://personality-project.org/r/r.commands.html>>

Display a string with variable inside:

cat(sprintf("<set name=\"%s\" value=\"%f\" ></set>\n", df$timeStamp, df$Price))

Pasted from <<http://stackoverflow.com/questions/3516008/how-to-print-r-variables-in-middle-of-string>>

> x <- 'say "Hello!"'  
> x  
[1] "say \"Hello!\""  
> cat(x)  
say "Hello!"

Pasted from <<http://stackoverflow.com/questions/15204442/how-to-display-text-with-quotes-in-r>>

x <- "say \"Hello!\""

R> cat(x)  
say "Hello!"

Pasted from <<http://stackoverflow.com/questions/15204442/how-to-display-text-with-quotes-in-r>>

# default is to use "fancy quotes"  
text <- c("check")  
message(dQuote(text))  
## “check”

# switch to straight quotes by setting an option  
options(useFancyQuotes = FALSE)  
message(dQuote(text))  
## "check"

# assign result to create a vector of quoted character strings  
text.quoted <- dQuote(text)  
message(text.quoted)  
## "check"

Pasted from <<http://stackoverflow.com/questions/15204442/how-to-display-text-with-quotes-in-r>>

Create a Diary or Log from Execution:

con <- file("test.log")  
sink(con, append=TRUE)  
sink(con, append=TRUE, type="message")

# This will echo all input and not truncate 150+ character lines...  
source("script.R", echo=TRUE, max.deparse.length=10000)

# Restore output to console  
sink()   
sink(type="message")

# And look at the log...  
cat(readLines("test.log"), sep="\n")

Pasted from <<http://stackoverflow.com/questions/7096989/how-to-save-all-console-output-to-file-in-r>>

Write into file:

write.matrix(x, file = "", sep = " ", blocksize)

Pasted from <<http://stat.ethz.ch/R-manual/R-patched/library/MASS/html/write.matrix.html>>

write(x, file = "data",  
 ncolumns = if(is.character(x)) 1 else 5,  
 append = FALSE, sep = " ")

Pasted from <<http://stat.ethz.ch/R-manual/R-devel/library/base/html/write.html>>

# create a 2 by 5 matrix  
x <- matrix(1:10, ncol = 5)

# the file data contains x, two rows, five cols  
# 1 3 5 7 9 will form the first row  
write(t(x))

# Writing to the "console" 'tab-delimited'  
# two rows, five cols but the first row is 1 2 3 4 5  
write(x, "", sep = "\t")  
unlink("data") # tidy up

Pasted from <<http://stat.ethz.ch/R-manual/R-devel/library/base/html/write.html>>

m <- matrix(1:12, 4 , 3)  
write.table(m,file="outfile,txt",sep="\t", col.names = F, row.names = F)

Pasted from <<http://stackoverflow.com/questions/10608526/writing-a-matrix-to-a-file-without-a-header-and-row-numbers>>

write.table(mtcars, file = "mtcars.txt", sep = " ")

Pasted from <<http://stackoverflow.com/questions/6957499/how-do-i-print-a-matrix-in-r>>

write.table(m, file = "m.txt", sep = " ", row.names = FALSE, col.names = FALSE)

Pasted from <<http://stackoverflow.com/questions/6957499/how-do-i-print-a-matrix-in-r>>

Save Matrix into CSV file:

write.matrix(format(moDat2, scientific=FALSE),   
 file = paste(targetPath, "dat2.csv", sep="/"), sep=",")

Pasted from <<http://stackoverflow.com/questions/13785303/save-matrix-to-csv-file-in-r-without-losing-format>>

mat <- matrix(1:10,ncol=2)  
rownames(mat) <- letters[1:5]  
colnames(mat) <- LETTERS[1:2]

mat  
write.table(mat,file="test.txt") # keeps the rownames  
read.table("test.txt",header=TRUE,row.names=1) # says first column are rownames  
unlink("test.txt")  
write.table(mat,file="test2.txt",row.names=FALSE) # drops the rownames  
read.table("test.txt",header=TRUE)   
unlink("test2.txt")

Pasted from <<http://stackoverflow.com/questions/6844166/export-matrix-in-r>>

Sparce Matrix:

library('Matrix')  
   
m1 <- matrix(0, nrow = 1000, ncol = 1000)  
m2 <- Matrix(0, nrow = 1000, ncol = 1000, sparse = TRUE)  
   
object.size(m1)  
# 8000200 bytes  
object.size(m2)  
# 5632 bytes

m1[500, 500] <- 1  
m2[500, 500] <- 1  
   
object.size(m1)  
# 8000200 bytes  
object.size(m2)  
# 5648 bytes

m2 %\*% rnorm(1000)  
m2 + m2  
m2 - m2  
t(m2)

m3 <- cBind(m2, m2)  
nrow(m3)  
ncol(m3)  
   
m4 <- rBind(m2, m2)  
nrow(m4)  
ncol(m4)

#sSecond Package Solution

library('slam')  
   
m1 <- matrix(0, nrow = 1000, ncol = 1000)  
m2 <- simple\_triplet\_zero\_matrix(nrow = 1000, ncol = 1000)  
   
object.size(m1)  
# 8000200 bytes  
object.size(m2)  
# 1032 bytes

# BUG HERE  
m1[500, 500] <- 1  
m2[500, 500] <- 1  
   
object.size(m1)  
object.size(m2)

m2 %\*% rnorm(1000)  
m2 + m2  
m2 - m2  
t(m2)

#Third Method:

library('Matrix')  
library('glmnet')  
   
n <- 10000  
p <- 500  
   
x <- matrix(rnorm(n \* p), n, p)  
iz <- sample(1:(n \* p),  
 size = n \* p \* 0.85,  
 replace = FALSE)  
x[iz] <- 0  
   
object.size(x)  
   
sx <- Matrix(x, sparse = TRUE)  
   
object.size(sx)  
   
beta <- rnorm(p)  
   
y <- x %\*% beta + rnorm(n)  
   
glmnet.fit <- glmnet(x, y)

#Fourth way that is more efficient

library('Matrix')  
library('glmnet')  
   
set.seed(1)  
performance <- data.frame()  
   
for (sim in 1:10)  
{  
 n <- 10000  
 p <- 500  
   
 nzc <- trunc(p / 10)  
   
 x <- matrix(rnorm(n \* p), n, p)  
 iz <- sample(1:(n \* p),  
 size = n \* p \* 0.85,  
 replace = FALSE)  
 x[iz] <- 0  
 sx <- Matrix(x, sparse = TRUE)  
   
 beta <- rnorm(nzc)  
 fx <- x[, seq(nzc)] %\*% beta  
   
 eps <- rnorm(n)  
 y <- fx + eps  
   
 sparse.times <- system.time(fit1 <- glmnet(sx, y))  
 full.times <- system.time(fit2 <- glmnet(x, y))  
 sparse.size <- as.numeric(object.size(sx))  
 full.size <- as.numeric(object.size(x))  
   
 performance <- rbind(performance, data.frame(Format = 'Sparse',  
 UserTime = sparse.times[1],  
 SystemTime = sparse.times[2],  
 ElapsedTime = sparse.times[3],  
 Size = sparse.size))  
 performance <- rbind(performance, data.frame(Format = 'Full',  
 UserTime = full.times[1],  
 SystemTime = full.times[2],  
 ElapsedTime = full.times[3],  
 Size = full.size))  
}  
   
ggplot(performance, aes(x = Format, y = UserTime, fill = Format)) +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'bar') +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'errorbar') +  
 ylab('User Time in Seconds') +  
 opts(legend.position = 'none')  
ggsave('sparse\_vs\_full\_user\_time.pdf')  
   
ggplot(performance, aes(x = Format, y = SystemTime, fill = Format)) +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'bar') +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'errorbar') +  
 ylab('System Time in Seconds') +  
 opts(legend.position = 'none')  
ggsave('sparse\_vs\_full\_system\_time.pdf')  
   
ggplot(performance, aes(x = Format, y = ElapsedTime, fill = Format)) +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'bar') +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'errorbar') +  
 ylab('Elapsed Time in Seconds') +  
 opts(legend.position = 'none')  
ggsave('sparse\_vs\_full\_elapsed\_time.pdf')  
   
ggplot(performance, aes(x = Format, y = Size / 1000000, fill = Format)) +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'bar') +  
 stat\_summary(fun.data = 'mean\_cl\_boot', geom = 'errorbar') +  
 ylab('Matrix Size in MB') +  
 opts(legend.position = 'none')  
ggsave('sparse\_vs\_full\_memory.pdf')

Pasted from <<http://www.johnmyleswhite.com/notebook/2011/10/31/using-sparse-matrices-in-r/>>

Optimization Function:

ans <- optimx(fn = function(x) sum(x\*x), par = 1:2)

coef(ans)

## Not run:

proj <- function(x) x/sum(x)

f <- function(x) -prod(proj(x))

ans <- optimx(1:2, f)

ans

coef(ans) <- apply(coef(ans), 1, proj)

ans

## End(Not run)

<http://cran.r-project.org/web/packages/optimx/optimx.pdf>

**Description**

Minimise a function subject to linear inequality constraints using an adaptive barrier algorithm.

**Usage**

constrOptim(theta, f, grad, ui, ci, mu = 1e-04, control = list(),  
 method = if(is.null(grad)) "Nelder-Mead" else "BFGS",  
 outer.iterations = 100, outer.eps = 1e-05, ...,  
 hessian = FALSE)

Pasted from <<http://stat.ethz.ch/R-manual/R-patched/library/stats/html/constrOptim.html>>

fr <- function(x) { x1 <- x[1]  
 x2 <- x[2]  
 -(log(x1) + x1^2/x2^2) # need negative since constrOptim is a minimization routine  
}

# define constraint

rbind(c(-1,-1),c(1,0), c(0,1) ) %\*% c(0.99,0.001) -c(-1,0, 0)

constrOptim(c(0.99,0.001), fr, NULL, ui=rbind(c(-1,-1), # the -x-y > -1  
 c(1,0), # the x > 0  
 c(0,1) ), # the y > 0  
 ci=c(-1,0, 0)) # the thresholds

#definegradiant for correct result

grr <- function(x) { ## Gradient of 'fr'  
 x1 <- x[1]  
 x2 <- x[2]  
 c(-(1/x[1] + 2 \* x[1]/x[2]^2),  
 2 \* x[1]^2 /x[2]^3 )  
}

constrOptim(c(0.99,0.001), fr, grr, ui=rbind(c(-1,-1), # the -x-y > -1  
 c(1,0), # the x > 0  
 c(0,1) ), # the y > 0  
 ci=c(-1,0, 0) )  
$par  
[1] 9.900007e-01 -3.542673e-16

$value  
[1] -7.80924e+30

$counts  
function gradient   
 2001 37

$convergence  
[1] 11

$message  
[1] "Objective function increased at outer iteration 2"

$outer.iterations  
[1] 2

$barrier.value  
[1] NaN

#another solution with better constraint

onstrOptim(c(0.25,0.25), fr, NULL,   
 ui=rbind( c(-1,-1), c(1,0), c(0,1) ),   
 ci=c(-1, 0.0001, 0.0001))   
$par

Pasted from <<http://stackoverflow.com/questions/5436630/constrained-optimization-in-r>>

Another Example:

## from optim  
fr <- function(x) { ## Rosenbrock Banana function  
 x1 <- x[1]  
 x2 <- x[2]  
 100 \* (x2 - x1 \* x1)^2 + (1 - x1)^2  
}  
grr <- function(x) { ## Gradient of 'fr'  
 x1 <- x[1]  
 x2 <- x[2]  
 c(-400 \* x1 \* (x2 - x1 \* x1) - 2 \* (1 - x1),  
 200 \* (x2 - x1 \* x1))  
}

optim(c(-1.2,1), fr, grr)  
#Box-constraint, optimum on the boundary  
constrOptim(c(-1.2,0.9), fr, grr, ui = rbind(c(-1,0), c(0,-1)), ci = c(-1,-1))  
# x <= 0.9, y - x > 0.1  
constrOptim(c(.5,0), fr, grr, ui = rbind(c(-1,0), c(1,-1)), ci = c(-0.9,0.1))

## Solves linear and quadratic programming problems  
## but needs a feasible starting value  
#  
# from example(solve.QP) in 'quadprog'  
# no derivative  
fQP <- function(b) {-sum(c(0,5,0)\*b)+0.5\*sum(b\*b)}  
Amat <- matrix(c(-4,-3,0,2,1,0,0,-2,1), 3, 3)  
bvec <- c(-8, 2, 0)  
constrOptim(c(2,-1,-1), fQP, NULL, ui = t(Amat), ci = bvec)  
# derivative  
gQP <- function(b) {-c(0, 5, 0) + b}  
constrOptim(c(2,-1,-1), fQP, gQP, ui = t(Amat), ci = bvec)

## Now with maximisation instead of minimisation  
hQP <- function(b) {sum(c(0,5,0)\*b)-0.5\*sum(b\*b)}  
constrOptim(c(2,-1,-1), hQP, NULL, ui = t(Amat), ci = bvec,  
 control = list(fnscale = -1))

Pasted from <<http://stat.ethz.ch/R-manual/R-patched/library/stats/html/constrOptim.html>>

BBML package:

start=list(mu=1,sigma=1) #starting values

mle.results<-mle2(norm.fit,start=list(mu=1,sigma=1),data=list(x)) #x is the name of the variable containing the data to be fitted

Pasted from <<http://www.pmc.ucsc.edu/~mclapham/Rtips/likelihood.htm>>

Parallel Processing:

library(parallel)

vignette(parallel)

Pasted from <<http://stackoverflow.com/questions/1395309/how-to-make-r-use-all-processors>>

Asample method for for loops:

library("parallel")  
library("foreach")  
library("doParallel")

cl <- makeCluster(detectCores() - 1)  
registerDoParallel(cl, cores = detectCores() - 1)

data = foreach(i = 1:length(filenames), .packages = c("ncdf","chron","stats"),  
 .combine = rbind) %dopar% {  
 try({  
 # your operations; line 1...  
 # your operations; line 2...  
 # your output  
 })  
}

Pasted from <<http://stackoverflow.com/questions/1395309/how-to-make-r-use-all-processors>>

> library(doMC)

> registerDoMC(cores = 5)

>

> ## All subsequent models are then run in parallel

> model <- train(y ~ ., data = training, method = "rf")

> gbmGrid <- expand.grid(.interaction.depth = c(1, 5, 9),

> .n.trees = (1:15)\*100,

> .shrinkage = 0.1)

In reality, train only created objects for 3 models and der

Pasted from <<http://caret.r-forge.r-project.org/parallel.html>>

To be read:

<http://cran.r-project.org/web/views/HighPerformanceComputing.html>

<http://caret.r-forge.r-project.org/parallel.html>

<http://www.r-bloggers.com/parallel-processing-when-does-it-worth/>

Maximum Likelihood example:

print(x) #print vector

hist(x) #histogram

dgamma(x, shape = alpha) #gamma distribution

dgamma(x, shape = alpha, log = TRUE) # log probability density rather than density

logl <- function(alpha, x)  
 return(sum(dgamma(x, shape = alpha, log = TRUE))) #calculate sum of likelihoods

logl <- function(alpha, x)  
 return(sum(dgamma(x, shape = alpha, log = TRUE))) #sum of likelihood

logl <- function(alpha, x) {  
 if (length(alpha) > 1) stop("alpha must be scalar")  
 if (alpha <= 0) stop("alpha must be positive")  
 return(sum(dgamma(x, shape = alpha, log = TRUE)))  
} #improved function

#function for doing log likelihood plot

logl <- function(alpha, x) {

if (length(alpha) < 1) stop("alpha must be scalar")

if (alpha <= 0) stop("alpha must be positive")

return(sum(dgamma(x, shape = alpha, log = TRUE)))

}

npoint <- 101

alphas <- seq(min(x), max(x), length = npoint)

logls <- double(npoint)

for (i in 1:npoint)

logls[i] <- logl(alphas[i], x)

plot(alphas, logls, type = "l",

xlab = expression(alpha), ylab = expression(l(alpha)))

#maximum likelihood: nlm, sample mean a good parameter estimator

mlogl <- function(alpha, x) {

if (length(alpha) < 1) stop("alpha must be scalar")

if (alpha <= 0) stop("alpha must be positive")

return(- sum(dgamma(x, shape = alpha, log = TRUE)))

}

out <- nlm(mlogl, mean(x), x = x)

print(out)

* + hessian returns the second derivative (an approximation calculated by finite differences) of the objective function. This will be a k × k matrix if the dimension of the parameter space is k.
  + fscale helps nlm know when it has converged to the solution. It should give roughly the size of the objective function at the solution. Often fscale = length(x) is about right.
  + print.level tells nlm to blather about what is is doing. print.level = 2 gives maximum verbosity.

mlogl <- function(alpha, x) {

if (length(alpha) < 1) stop("alpha must be scalar")

if (alpha <= 0) stop("alpha must be positive")

return(- sum(dgamma(x, shape = alpha, log = TRUE)))

}

out <- nlm(mlogl, mean(x), x = x, hessian = TRUE,

fscale = length(x), print.level = 2)

print(out)

#asymptotic confidence interval: Fisher information and confidence interval

conf.level <- 0.95

mlogl <- function(alpha, x) {

if (length(alpha) < 1) stop("alpha must be scalar")

if (alpha <= 0) stop("alpha must be positive")

return(- sum(dgamma(x, shape = alpha, log = TRUE)))

}

n <- length(x)

out <- nlm(mlogl, mean(x), x = x, hessian = TRUE,

fscale = n)

alpha.hat <- out$estimate

z <- qnorm((1 + conf.level) / 2)

# confidence interval using expected Fisher information

alpha.hat + c(-1, 1) \* z / sqrt(n \* trigamma(alpha.hat))

# confidence interval using observed Fisher information

alpha.hat + c(-1, 1) \* z / sqrt(out$hessian)

#several parameters

mlogl <- function(theta, x) {  
 if (length(theta) != 2) stop("theta must be vector of length 2")  
 alpha <- theta[1]  
 lambda <- theta[2]  
 if (alpha <= 0) stop("theta[1] must be positive")  
 if (lambda <= 0) stop("theta[2] must be positive")  
 return(- sum(dgamma(x, shape = alpha, rate = lambda, log = TRUE)))  
}

alpha.start <- mean(x)^2 / var(x)  
lambda.start <- mean(x) / var(x)

mlogl <- function(theta, x) {

if (length(theta) != 2)

stop("theta must be vector of length 2")

alpha <- theta[1]

lambda <- theta[2]

if (alpha <= 0) stop("theta[1] must be positive")

if (lambda <= 0) stop("theta[2] must be positive")

return(- sum(dgamma(x, shape = alpha, rate = lambda,

log = TRUE)))

}

alpha.start <- mean(x)^2 / var(x)

lambda.start <- mean(x) / var(x)

theta.start <- c(alpha.start, lambda.start)

out <- nlm(mlogl, theta.start, x = x, hessian = TRUE,

fscale = length(x))

print(out)

print(theta.start)

eigen(out$hessian, symmetric = TRUE, only.values = TRUE)

# theoretical Fisher information

n <- length(x)

alpha.hat <- out$estimate[1]

lambda.hat <- out$estimate[2]

fish <- n \* matrix(c(trigamma(alpha.hat), - 1 / lambda.hat,

- 1 / lambda.hat, alpha.hat / lambda.hat^2), nrow = 2)

fish

conf.level <- 0.95

mlogl <- function(theta, x) {

if (length(theta) != 2)

stop("theta must be vector of length 2")

alpha <- theta[1]

lambda <- theta[2]

if (alpha <= 0) stop("theta[1] must be positive")

if (lambda <= 0) stop("theta[2] must be positive")

return(- sum(dgamma(x, shape = alpha, rate = lambda,

log = TRUE)))

}

alpha.start <- mean(x)^2 / var(x)

lambda.start <- mean(x) / var(x)

theta.start <- c(alpha.start, lambda.start)

out <- nlm(mlogl, theta.start, x = x, hessian = TRUE,

fscale = length(x))

crit.val <- qnorm((1 + conf.level) / 2)

inv.fish.info <- solve(out$hessian)

for (i in 1:2)

print(out$estimate[i] + c(-1, 1) \* crit.val \*

sqrt(inv.fish.info[i, i]))

#example of A five-parameter Normal Mixture Example

length(x)

summary(x)

hist(x)

mlogl <- function(theta) {  
 stopifnot(is.numeric(theta))  
 stopifnot(length(theta) == 5)  
 mu1 <- theta[1]  
 mu2 <- theta[2]  
 v1 <- theta[3]  
 v2 <- theta[4]  
 p <- theta[5]  
 logl <- sum(log(p \* dnorm(x, mu1, sqrt(v1)) +  
 (1 - p) \* dnorm(x, mu2, sqrt(v2))))  
 return(- logl)  
}

#Maximum Likelihood

mlogl <- function(theta) {

stopifnot(is.numeric(theta))

stopifnot(length(theta) == 5)

mu1 <- theta[1]

mu2 <- theta[2]

v1 <- theta[3]

v2 <- theta[4]

p <- theta[5]

logl <- sum(log(p \* dnorm(x, mu1, sqrt(v1)) +

(1 - p) \* dnorm(x, mu2, sqrt(v2))))

return(- logl)

}

n <- length(x)

p.start <- 1 / 2

mu1.start <- mean(sort(x)[seq(along = x) <= n / 2])

mu2.start <- mean(sort(x)[seq(along = x) > n / 2])

v1.start <- var(sort(x)[seq(along = x) <= n / 2])

v2.start <- var(sort(x)[seq(along = x) > n / 2])

theta.start <- c(mu1.start, mu2.start, v1.start,

v2.start, p.start)

out <- nlm(mlogl, theta.start, print.level = 2,

fscale = length(x))

out <- nlm(mlogl, out$estimate, print.level = 2,

fscale = length(x), hessian = TRUE)

print(out)

print(theta.start)

mu1.hat <- out$estimate[1]

mu2.hat <- out$estimate[2]

sigma1.hat <- sqrt(out$estimate[3])

sigma2.hat <- sqrt(out$estimate[4])

p.hat <- out$estimate[5]

fred <- function(x) p.hat \* dnorm(x, mu1.hat, sigma1.hat) +

(1 - p.hat) \* dnorm(x, mu2.hat, sigma2.hat)

hist(x, freq = FALSE)

curve(fred, add = TRUE)

hist(x, freq = FALSE,

ylim = range(0, fred(mu1.hat), fred(mu2.hat)))

curve(fred, add = TRUE)

curve(p.hat \* dnorm(x, mu1.hat, sigma1.hat),

add = TRUE, col = "red")

curve((1 - p.hat) \* dnorm(x, mu2.hat, sigma2.hat),

add = TRUE, col = "red")

eigen(out$hessian, symmetric = TRUE)

#Confidence Interval

conf.level <- 0.95

mlogl <- function(theta) {

stopifnot(is.numeric(theta))

stopifnot(length(theta) == 5)

mu1 <- theta[1]

mu2 <- theta[2]

v1 <- theta[3]

v2 <- theta[4]

p <- theta[5]

logl <- sum(log(p \* dnorm(x, mu1, sqrt(v1)) +

(1 - p) \* dnorm(x, mu2, sqrt(v2))))

return(- logl)

}

n <- length(x)

p.start <- 1 / 2

mu1.start <- mean(sort(x)[seq(along = x) <= n / 2])

mu2.start <- mean(sort(x)[seq(along = x) > n / 2])

v1.start <- var(sort(x)[seq(along = x) <= n / 2])

v2.start <- var(sort(x)[seq(along = x) > n / 2])

theta.start <- c(mu1.start, mu2.start, v1.start,

v2.start, p.start)

out <- nlm(mlogl, theta.start, fscale = length(x))

out <- nlm(mlogl, out$estimate,

fscale = length(x), hessian = TRUE)

crit.val <- qnorm((1 + conf.level) / 2)

inv.fish.info <- solve(out$hessian)

for (i in 1:length(out$estimate))

print(out$estimate[i] + c(-1, 1) \* crit.val \*

sqrt(inv.fish.info[i, i]))

Pasted from <<http://www.stat.umn.edu/geyer/5102/examp/like.html>>

There are a number of general-purpose optimization routines in base R that I'm aware of: optim,nlminb, nlm and constrOptim (which handles linear inequality constraints, and calls optim under the hood). Here are some things that you might want to consider in choosing which one to use.

* optim can use a number of different algorithms including conjugate gradient, Newton, quasi-Newton, Nelder-Mead and simulated annealing. The last two don't need gradient information and so can be useful if gradients aren't available or not feasible to calculate (but are likely to be slower and require more parameter fine-tuning, respectively). It also has an option to return the computed Hessian at the solution, which you would need if you want standard errors along with the solution itself.

which is probably the most-used optimizer, provides a few different optimization routines; for example, BFGS, L-BFGS-B, and simulated annealing (via the SANN option),

the latter of which might be handy if you have a difficult optimizing problem.

* nlminb uses a quasi-Newton algorithm that fills the same niche as the "L-BFGS-B" method inoptim. In my experience it seems a bit more robust than optim in that it's more likely to return a solution in marginal cases where optim will fail to converge, although that's likely problem-dependent. It has the nice feature, if you provide an explicit gradient function, of doing a numerical check of its values at the solution. If these values don't match those obtained from numerical differencing, nlminb will give a warning; this helps to ensure you haven't made a mistake in specifying the gradient (easy to do with complicated likelihoods).

Provides a way to constrain parameter values to particular bounding boxes

* nlm only uses a Newton algorithm. This can be faster than other algorithms in the sense of needing fewer iterations to reach convergence, but has its own drawbacks. It's more sensitive to the shape of the likelihood, so if it's strongly non-quadratic, it may be slower or you may get convergence to a false solution. The Newton algorithm also uses the Hessian, and computing that can be slow enough in practice that it more than cancels out any theoretical speedup.

will work just fine if the likelihood surface isn't particularly "rough" and is everywhere differentiable

* rgenoud, for instance, provides a genetic algorithm for optimization. Genetic algorithms can be slow to converge, but are usually guaranteed to converge (in time) even when there are discontinuities in the likelihood.  is set up to use snow for parallel processing, which helps somewhat.

<http://sekhon.berkeley.edu/rgenoud/>

* DEoptim uses a different genetic optimization routine

Pasted from <<http://stats.stackexchange.com/questions/9535/when-should-i-not-use-rs-nlm-function-for-mle>>

Very importance Source:

<http://cran.r-project.org/web/views/Optimization.html>

* Optimplex: Provides a building block for optimization algorithms

based on a simplex. The optimsimplex package may be used in the

following optimization methods: the simplex method of Spendley

et al., the method of Nelder and Mead, Box’s algorithm for

constrained optimization, the multi-dimensional search by Torczon, etc…

<http://cran.r-project.org/web/packages/optimsimplex/optimsimplex.pdf>

<http://cran.r-project.org/web/packages/optimsimplex/index.html>

Other Optimization nonlinear Algorithms:

<http://cran.r-project.org/web/packages/nloptr/nloptr.pdf>

<http://cran.r-project.org/web/packages/alabama/alabama.pdf>

NLM Example:

**Examples**

f <- function(x) sum((x-1:length(x))^2)  
nlm(f, c(10,10))  
nlm(f, c(10,10), print.level = 2)  
utils::str(nlm(f, c(5), hessian = TRUE))

f <- function(x, a) sum((x-a)^2)  
nlm(f, c(10,10), a = c(3,5))  
f <- function(x, a)  
{  
 res <- sum((x-a)^2)  
 attr(res, "gradient") <- 2\*(x-a)  
 res  
}  
nlm(f, c(10,10), a = c(3,5))

## more examples, including the use of derivatives.  
## Not run: demo(nlm)

Pasted from <<http://stat.ethz.ch/R-manual/R-devel/library/stats/html/nlm.html>>

Univar ate Binomial and multinomial inference:

> x <- c(12, 8, 10)

> p <- c(0.4, 0.3, 0.3)

> chisq.test(x, p=p)

Chi-squared test for given probabilities

data: x

X-squared = 0.2222, df = 2, p-value = 0.8948

> chisq.test(x, p=p, simulate.p.value=TRUE, B=10000)

Chi-squared test for given probabilities with

simulated p-value (based on 10000 replicates)

data: x

X-squared = 0.2222, df = NA, p-value = 0.8763

Bayesian Inference:

library("TeachingDemos")

y <- 0; n <- 25

a1 <- 3.6; a2 <- 41.4

a <- a1 + y; b <- a2 + n

h <- hpd(qbeta, shape1=a, shape2=b)

Two way continuity table:

> x<- c(9,8,27,8,47,236,23,39,88,49,179,706,28,48,89,19,104,293)

> data <- matrix(x, nrow=3,ncol=6, byrow=TRUE)

> dimnames(data) = list(Degree=c("< HS","HS","College"),Belief=c("1","2","3","4","5","6"))

> install.packages("vcdExtra")

> library("vcdExtra")

> StdResid <- c(-0.4,-2.2,-1.4,-1.5,-1.3,3.6,-2.5,-2.6,-3.3,1.8,0.0,3.4,3.1,4.7,4.8,-0.8,1.1,-6.7)

> StdResid <- matrix(StdResid,nrow=3,ncol=6,byrow=TRUE)

> mosaic(data,residuals = StdResid, gp=shading\_Friendly)

Chi-Square and Fisher's exact test; Residuals:

> data <- matrix(c(9,8,27,8,47,236,23,39,88,49,179,706,28,48,89,19,104,293),

ncol=6,byrow=TRUE)

> chisq.test(data)

Pearson’s Chi-squared test

data: data

X-squared = 76.1483, df = 10, p-value = 2.843e-12

> chisq.test(data)$stdres

[,1] [,2] [,3] [,4] [,5] [,6]

[1,] -0.368577 -2.227511 -1.418621 -1.481383 -1.3349600 3.590075

[2,] -2.504627 -2.635335 -3.346628 1.832792 0.0169276 3.382637

[3,] 3.051857 4.724326 4.839597 -0.792912 1.0794638 -6.665195

> yes <- c(54,25)

> n <- c(10379,51815)

> x <- c(1,0)

> fit <- glm(yes/n ~ x, weights=n, family=binomial(link=logit))

> summary(fit)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -7.6361 0.2000 -38.17 <2e-16 \*\*\*

x 2.3827 0.2421 9.84 <2e-16 \*\*\*

>

confint(fit)

Waiting for profiling to be done...

2.5 % 97.5 %

(Intercept) -8.055554 -7.268025

x 1.919634 2.873473

> exp(1.919634); exp(2.873473)

[1] 6.818462

[1] 17.69838

> tea <- matrix(c(3,1,1,3),ncol=2,byrow=TRUE)

> fisher.test(tea)

> fisher.test(table, simulate.p.value=TRUE, B=10000)

Generalized Linear Models:

> snoring <- matrix(c(24,1355,35,603,21,192,30,224), ncol=2, byrow=TRUE)

> scores <- c(0,2,4,5)

> snoring.fit <- glm(snoring ~ scores, family=binomial(link=logit))

> summary(snoring.fit)

Call:

glm(formula = snoring ~ scores, family = binomial(link = logit))

Deviance Residuals:

1 2 3 4

-0.8346 1.2521 0.2758 -0.6845

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -3.86625 0.16621 -23.261 < 2e-16 \*\*\*

scores 0.39734 0.05001 7.945 1.94e-15 \*\*\*

--Signif.

codes: 0 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 65.9045 on 3 degrees of freedom

Residual deviance: 2.8089 on 2 degrees of freedom

AIC: 27.061

Number of Fisher Scoring iterations: 4

> pearson <- summary.lm(snoring.fit)$residuals # Pearson residuals

> hat <- lm.influence(snoring.fit)$hat # hat or leverage values

> stand.resid <- pearson/sqrt(1 - hat) # standardized residuals

> cbind(scores, snoring, fitted(snoring.fit), pearson, stand.resid)

scores pearson stand.resid

1 0 24 1355 0.02050742 -0.8131634 -1.6783847

2 2 35 603 0.04429511 1.2968557 1.5448873

3 4 21 192 0.09305411 0.2781891 0.3225535

4 5 30 224 0.13243885 -0.6736948 -1.1970179

> fit <- glm(y ~ x, family=quasi(variance="mu(1-mu)"),start=c(0.5, 0))

> summary(fit, dispersion=1)

> crabs <- read.table("crab.dat",header=T)

> crabs

color spine width satellites weight

1 3 3 28.3 8 3050

2 4 3 22.5 0 1550

3 2 1 26.0 9 2300

4 4 3 24.8 0 2100

5 4 3 26.0 4 2600

6 3 3 23.8 0 2100

....

173 3 2 24.5 0 2000

> weight <- weight/1000 # weight in kilograms rather than grams

> fit <- glm(satellites ~ weight, family=poisson(link=log), data=crabs)

> summary(fit)

> library(MASS)

> fit.nb <- glm.nb(satell ~ weight, link=log)

> summary(fit.nb)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -0.8647 0.4048 -2.136 0.0327 \*

weight2 0.7603 0.1578 4.817 1.45e-06 \*\*\*

---

Null deviance: 216.43 on 172 degrees of freedom

Residual deviance: 196.16 on 171 degrees of freedom

AIC: 754.64

Theta: 0.931

Std. Err.: 0.168

2 x log-likelihood: -748.644

> fit <- glm(... model formula, family, data, etc ...)

> rstandard(fit, type="pearson")

# This borrows heavily from Laura Thompson’s manual at

# <https://home.comcast.net/~lthompson221/Splusdiscrete2.pdf>

> rats <- read.table("teratology.dat", header = T)

> rats # Full data set of 58 litters at course website

litter group n y

1 1 1 10 1

2 2 1 11 4

3 3 1 12 9

57 57 4 6 0

58 58 4 17 0

> rats$group <- as.factor(rats$group)

> fit.bin <- glm(y/n ~ group - 1, weights = n, data=rats, family=binomial)

> summary(fit.bin)

Coefficients: # these are the sample logits

Estimate Std. Error z value Pr(>|z|)

group1 1.1440 0.1292 8.855 < 2e-16 \*\*\*

group2 -2.1785 0.3046 -7.153 8.51e-13 \*\*\*

group3 -3.3322 0.7196 -4.630 3.65e-06 \*\*\*

group4 -2.9857 0.4584 -6.514 7.33e-11 \*\*\*

---

Null deviance: 518.23 on 58 degrees of freedom

Residual deviance: 173.45 on 54 degrees of freedom

AIC: 252.92

> (pred <- unique(predict(fit.bin, type="response")))

[1] 0.75840979 0.10169492 0.03448276 0.04807692 # sample proportions

> (SE <- sqrt(pred\*(1-pred)/tapply(rats$n,rats$group,sum)))

1 2 3 4

0.02367106 0.02782406 0.02395891 0.02097744 # SE’s of proportions

> (X2 <- sum(resid(fit.bin, type="pearson")^2)) # Pearson stat.

[1] 154.707

> phi <- X2/(58 - 4) # estimate of phi for QL analysis

> phi

[1] 2.864945

> SE\*sqrt(phi)

1 2 3 4

0.04006599 0.04709542 0.04055320 0.03550674 # adjusted SE’s for proportions

> fit.ql <- glm(y/n ~ group - 1, weights=n, data=rats, family=quasi(link=identity,

variance="mu(1-mu)"),start=unique(predict(fit.bin,type="response")))

> summary(fit.ql) # This shows another way to get the QL results

Coefficients:

Estimate Std. Error t value Pr(>|t|)

group1 0.75841 0.04007 18.929 <2e-16 \*\*\*

group2 0.10169 0.04710 2.159 0.0353 \*

group3 0.03448 0.04055 0.850 0.3989

group4 0.04808 0.03551 1.354 0.1814

--(Dispersion

parameter for quasi family taken to be 2.864945)

Logistic Regression

> crabs <- read.table("crabs.dat",header=TRUE)

> crabs

color spine width satellites weight

1 3 3 28.3 8 3050

2 4 3 22.5 0 1550

3 2 1 26.0 9 2300

....

173 3 2 24.5 0 2000

> y <- ifelse(crabs$satellites > 0, 1, 0) # y = a binary indicator of satellites

> crabs$weight <- crabs$weight/1000 # weight in kilograms rather than grams

> fit <- glm(y ~ weight, family=binomial(link=logit), data=crabs)

> summary(fit)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -3.6947 0.8802 -4.198 2.70e-05 \*\*\*

weight 1.8151 0.3767 4.819 1.45e-06 \*\*\*

---

Null Deviance: 225.7585 on 172 degrees of freedom

Residual Deviance: 195.7371 on 171 degrees of freedom

AIC: 199.74

> crabs$color <- crabs$color - 1 # color now takes values 1,2,3,4

> crabs$color <- factor(crabs$color) # treat color as a factor

> fit2 <- glm(y ~ weight + color, family=binomial(link=logit), data=crabs)

> summary(fit2)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -3.2572 1.1985 -2.718 0.00657 \*\*

weight 1.6928 0.3888 4.354 1.34e-05 \*\*\*

color2 0.1448 0.7365 0.197 0.84410

color3 -0.1861 0.7750 -0.240 0.81019

color4 -1.2694 0.8488 -1.495 0.13479

---

(Dispersion Parameter for Binomial family taken to be 1 )

Null Deviance: 225.7585 on 172 degrees of freedom

Residual Deviance: 188.5423 on 168 degrees of freedom

AIC: 198.54

> yes <- c(24,35,21,30)

> n <- c(1379,638,213,254)

> scores <- c(0,2,4,5)

> fit <- glm(yes/n ~ scores, weights=n, family=binomial(link=logit))

> fit

Coefficients:

(Intercept) scores

-3.8662 0.3973

Degrees of Freedom: 3 Total (i.e. Null); 2 Residual

Null Deviance: 65.9

Residual Deviance: 2.809 AIC: 27.06

ROC curves:

> dose <- c(rep(1.691,59),rep(1.724,60),rep(1.755,62),rep(1.784,56),

+ rep(1.811,63),rep(1.837,59),rep(1.861,62),rep(1.884,60))

> y <- c(rep(1,6),rep(0,53),rep(1,13),rep(0,47),rep(1,18),rep(0,44),

+ rep(1,28),rep(0,28),rep(1,52),rep(0,11),rep(1,53),rep(0,6),

+ rep(1,61),rep(0,1),rep(1,60))

> fit.probit <- glm(y ~ dose, family=binomial(link=probit))

> summary(fit.probit)

Estimate Std. Error z value Pr(>|z|)

(Intercept) -34.956 2.649 -13.20 <2e-16

dose 19.741 1.488 13.27 <2e-16

---

> library("ROCR") # to construct ROC curve

> pred <- prediction(fitted(fit.probit),y)

> perf <- performance(pred, "tpr", "fpr")

> plot(perf)

> performance(pred,"auc")

Slot "y.values":

[[1]]

[1] 0.9010852 # area under ROC curve

Cochran-Mantel-Haenszel test:

> beitler <- c(11,10,25,27,16,22,4,10,14,7,5,12,2,1,14,16,6,0,11,12,1,0,10,10,1,1,4,8,4,6,2,1)

> beitler <- array(beitler, dim=c(2,2,8))

> mantelhaen.test(beitler, correct=FALSE)

Mantel-Haenszel chi-squared test without continuity correction

data: beitler

Mantel-Haenszel X-squared = 6.3841, df = 1, p-value = 0.01151

alternative hypothesis: true common odds ratio is not equal to 1

95 percent confidence interval:

1.177590 3.869174

sample estimates:

common odds ratio

2.134549

> mantelhaen.test(beitler, correct=FALSE, exact=TRUE)

Other Binary Response Models:

> fit.probit <- glm(y ~ weight, family=binomial(link=probit), data=crabs)

> summary(fit.probit)

Coefficients:

Value Std. Error t value

(Intercept) -2.238245 0.5114850 -4.375974

weight 1.099017 0.2150722 5.109989

Residual Deviance: 195.4621 on 171 degrees of freedom

> beetles <- read.table("beetle.dat", header=T)

> beetles

dose number killed

1 1.691 59 6

2 1.724 60 13

3 1.755 62 18

4 1.784 56 28

5 1.811 63 52

6 1.837 59 53

7 1.861 62 61

8 1.884 60 60

> binom.dat <- matrix(append(killed,number-killed),ncol=2)

> fit.cloglog <- glm(binom.dat ~ dose, family=binomial(link=cloglog),

data=beetles)

> summary(fit.cloglog) # much better fit than logit

Value Std. Error t value

(Intercept) -39.52250 3.232269 -12.22748

dose 22.01488 1.795086 12.26397

Null Deviance: 284.2024 on 7 degrees of freedom

Residual Deviance: 3.514334 on 6 degrees of freedom

> pearson.resid <- resid(fit.cloglog, type="pearson")

> std.resid <- pearson.resid/sqrt(1-lm.influence(fit.cloglog)$hat)

> cbind(dose, killed/number, fitted(fit.cloglog), pearson.resid, std.resid)

dose pearson.resid std.resid

1 1.691 0.1016949 0.09582195 0.1532583 0.1772659

2 1.724 0.2166667 0.18802653 0.5677671 0.6694966

3 1.755 0.2903226 0.33777217 -0.7899738 -0.9217717

4 1.784 0.5000000 0.54177644 -0.6274464 -0.7041154

5 1.811 0.8253968 0.75683967 1.2684541 1.4855799

6 1.837 0.8983051 0.91843509 -0.5649292 -0.7021989

7 1.861 0.9838710 0.98575181 -0.1249636 -0.1489834

8 1.884 1.0000000 0.99913561 0.2278334 0.2368981

> confint(fit.cloglog)

2.5 % 97.5 %

(Intercept) -46.13984 -33.49923

dose 18.66945 25.68877

Penalized Likelihood

> x <- c(12, 15, 42, 52, 59, 73, 82, 91, 96, 105, 114, 120, 121, 128, 130,

139, 139, 157, 1, 1, 2, 8, 11, 18, 22, 31, 37, 61, 72, 81, 97,

112, 118, 127, 131, 140, 151, 159, 177, 206)

> y <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0)

> k1 <- ksmooth(x,y,"normal",bandwidth=25)

> k2 <- ksmooth(x,y,"normal",bandwidth=100)

> plot(x,y)

> lines(k1)

> lines(k2, lty=2)

Generalized Additive Model:

> library(vgam)

> gam.fit <- vgam(y ~ s(weight), family=binomialff(link=logit), data=crabs)

> plot(weight, fitted(gam.fit))

Mutlinomial Response Models

> alligators <- read.table("alligators.dat",header=TRUE)

> alligators

lake gender size y1 y2 y3 y4 y5

1 1 1 1 7 1 0 0 5

2 1 1 0 4 0 0 1 2

3 1 0 1 16 3 2 2 3

4 1 0 0 3 0 1 2 3

5 2 1 1 2 2 0 0 1

6 2 1 0 13 7 6 0 0

7 2 0 1 0 1 0 1 0

8 2 0 0 3 9 1 0 2

9 3 1 1 3 7 1 0 1

10 3 1 0 8 6 6 3 5

11 3 0 1 2 4 1 1 4

12 3 0 0 0 1 0 0 0

13 4 1 1 13 10 0 2 2

14 4 1 0 9 0 0 1 2

15 4 0 1 3 9 1 0 1

16 4 0 0 8 1 0 0 1

> library(VGAM)

> vglm(formula = cbind(y2,y3,y4,y5,y1) ~ size + factor(lake),

family=multinomial, data=alligators)

Coefficients:

(Intercept):1 (Intercept):2 (Intercept):3 (Intercept):4 size:1

-3.2073772 -2.0717560 -1.3979592 -1.0780754 1.4582046

size:2 size:3 size:4 factor(lake)2:1 factor(lake)2:2

-0.3512628 -0.6306597 0.3315503 2.5955779 1.2160953

factor(lake)2:3 factor(lake)2:4 factor(lake)3:1 factor(lake)3:2 factor(lake)3:3

-1.3483253 -0.8205431 2.7803434 1.6924767 0.3926492

factor(lake)3:4 factor(lake)4:1 factor(lake)4:2 factor(lake)4:3 factor(lake)4:4

0.6901725 1.6583586 -1.2427766 -0.6951176 -0.8261962

Degrees of Freedom: 64 Total; 44 Residual

Residual Deviance: 52.47849

Log-likelihood: -74.42948

> library(nnet)

> fit2 <- multinom(cbind(y1,y2,y3,y4,y5) ~ size + factor(lake), data=alligators)

> summary(fit2)

Call:

multinom(formula = cbind(y1, y2, y3, y4, y5) ~ size + factor(lake),

data = alligators)

Coefficients:

(Intercept) size factor(lake)2 factor(lake)3 factor(lake)4

y2 -3.207394 1.4582267 2.5955898 2.7803506 1.6583514

y3 -2.071811 -0.3512070 1.2161555 1.6925186 -1.2426769

y4 -1.397976 -0.6306179 -1.3482294 0.3926516 -0.6951107

y5 -1.078137 0.3315861 -0.8204767 0.6902170 -0.8261528

Std. Errors:

(Intercept) size factor(lake)2 factor(lake)3 factor(lake)4

y2 0.6387317 0.3959455 0.6597077 0.6712222 0.6128757

y3 0.7067258 0.5800273 0.7860141 0.7804482 1.1854024

y4 0.6085176 0.6424744 1.1634848 0.7817677 0.7812585

y5 0.4709212 0.4482539 0.7296253 0.5596752 0.5575414

Residual Deviance: 540.0803

AIC: 580.0803

VGLM for Ordinal Models

> happy <- read.table("happy.dat", header=TRUE)

> happy

race traumatic y1 y2 y3

1 0 0 1 0 0

2 0 0 1 0 0

3 0 0 1 0 0

4 0 0 1 0 0

5 0 0 1 0 0

6 0 0 1 0 0

7 0 0 1 0 0

8 0 0 0 1 0

...

94 1 2 0 0 1

95 1 3 0 1 0

96 1 3 0 1 0

97 1 3 0 0 1

> library(VGAM)

> fit <- vglm(cbind(y1,y2,y3) ~ race + traumatic,

family=cumulative(parallel=TRUE), data=happy)

> summary(fit)

Coefficients:

Value Std. Error t value

(Intercept):1 -0.51812 0.33819 -1.5320

(Intercept):2 3.40060 0.56481 6.0208

race -2.03612 0.69113 -2.9461

traumatic -0.40558 0.18086 -2.2425

Names of linear predictors: logit(P[Y<=1]), logit(P[Y<=2])

Residual Deviance: 148.407 on 190 degrees of freedom

Log-likelihood: -74.2035 on 190 degrees of freedom

Number of Iterations: 5

> fit.inter <- vglm(cbind(y1,y2,y3) ~ race + traumatic + race\*traumatic,

family=cumulative(parallel=TRUE), data=happy)

> summary(fit.inter)

Coefficients:

Value Std. Error t value

(Intercept):1 -0.43927 0.34469 -1.2744

(Intercept):2 3.52745 0.58737 6.0055

race -3.05662 1.20459 -2.5375

traumatic -0.46905 0.19195 -2.4436

race:traumatic 0.60850 0.60077 1.0129

Residual Deviance: 147.3575 on 189 degrees of freedom

Log-likelihood: -73.67872 on 189 degrees of freedom

Number of Iterations: 5

> fit2 <- vglm(cbind(y1,y2,y3) ~ race + traumatic, family=cumulative,

data=happy)

> summary(fit2)

Coefficients:

Value Std. Error t value

(Intercept):1 -0.56605 0.36618 -1.545821

(Intercept):2 3.48370 0.75950 4.586850

race:1 -14.01877 322.84309 -0.043423

race:2 -1.84673 0.76276 -2.421095

traumatic:1 -0.34091 0.21245 -1.604644

traumatic:2 -0.48356 0.27524 -1.756845

Residual Deviance: 146.9951 on 188 degrees of freedom

Log-likelihood: -73.49755 on 188 degrees of freedom

Number of Iterations: 14

> pchisq(deviance(fit)-deviance(fit2),df=df.residual(fit)-df.residual(fit2),lower.tail=FALSE)

[1] 0.4936429

fit.probit <- vglm(cbind(y1,y2,y3) ~ race + traumatic,

family=cumulative(link=probit, parallel=TRUE), data=happy)

> summary(fit.probit)

Coefficients:

Value Std. Error t value

(Intercept):1 -0.34808 0.200147 -1.7391

(Intercept):2 1.91607 0.282872 6.7736

race -1.15712 0.378716 -3.0554

traumatic -0.22131 0.098973 -2.2361

Residual Deviance: 148.1066 on 190 degrees of freedom

Log-likelihood: -74.0533 on 190 degrees of freedom

Number of Iterations: 5

To ﬁt the adjacent-categories logit model to the same data, we use

> fit.acat <- vglm(cbind(y1,y2,y3) ~ race + traumatic,

family=acat(reverse=TRUE, parallel=TRUE), data=happy)

> summary(fit.acat)

Coefficients:

Value Std. Error t value

(Intercept):1 -0.49606 0.31805 -1.5597

(Intercept):2 3.02747 0.57392 5.2751

race -1.84230 0.64190 -2.8701

traumatic -0.35701 0.16396 -2.1775

Names of linear predictors: log(P[Y=1]/P[Y=2]), log(P[Y=2]/P[Y=3])

Residual Deviance: 148.1996 on 190 degrees of freedom

Log-likelihood: -74.09982 on 190 degrees of freedom

Number of Iterations: 5

> fit.cratio <- vglm(cbind(y1,y2,y3) ~ race + traumatic,

family=cratio(reverse=TRUE, parallel=TRUE), data=happy)

> summary(fit.cratio)

Coefficients:

Value Std. Error t value

(Intercept):1 -0.45530 0.32975 -1.3808

(Intercept):2 3.34108 0.56309 5.9335

race -2.02555 0.67683 -2.9927

traumatic -0.38504 0.17368 -2.2170

Names of linear predictors: logit(P[Y<2|Y<=2]), logit(P[Y<3|Y<=3])

Residual Deviance: 148.1571 on 190 degrees of freedom

Log-likelihood: -74.07856 on 190 degrees of freedom

Number of Iterations: 5

Other Multinomial Functions:

> library(MASS)

> response <- matrix(0,nrow=97,ncol=1)

> response <- ifelse(y1==1,1,0)

> response <- ifelse(y2==1,2,resp)

> response <- ifelse(y3==1,3,resp)

> y <- factor(response)

> polr(y ~ race + traumatic, data=happy)

Call:

polr(formula = y ~ race + traumatic, data=happy)

Coefficients:

race traumatic

2.0361187 0.4055724

Intercepts:

1|2 2|3

-0.5181118 3.4005955

Residual Deviance: 148.407

AIC: 156.407

Loglinear Models:

> drugs <- read.table("drugs.dat",header=TRUE)

> drugs

a c m count

1 yes yes yes 911

2 yes yes no 538

3 yes no yes 44

4 yes no no 456

5 no yes yes 3

6 no yes no 43

7 no no yes 2

8 no no no 279

> alc <- factor(a); cig <- factor(c); mar <- factor(m)

> indep <- glm(count ~ alc + cig + mar, family=poisson(link=log), data=drugs)

> summary(indep) % loglinear model (A, C, M)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 6.29154 0.03667 171.558 < 2e-16 \*\*\*

alc2 -1.78511 0.05976 -29.872 < 2e-16 \*\*\*

cig2 -0.64931 0.04415 -14.707 < 2e-16 \*\*\*

mar2 0.31542 0.04244 7.431 1.08e-13 \*\*\*

---

Null deviance: 2851.5 on 7 degrees of freedom

Residual deviance: 1286.0 on 4 degrees of freedom

AIC: 1343.1

Number of Fisher Scoring iterations: 6

> homo.assoc <- update(indep, .~. + alc:cig + alc:mar + cig:mar)

> summary(homo.assoc) # loglinear model (AC, AM, CM)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 6.81387 0.03313 205.699 < 2e-16 \*\*\*

alc2 -5.52827 0.45221 -12.225 < 2e-16 \*\*\*

cig2 -3.01575 0.15162 -19.891 < 2e-16 \*\*\*

mar2 -0.52486 0.05428 -9.669 < 2e-16 \*\*\*

alc2:cig2 2.05453 0.17406 11.803 < 2e-16 \*\*\*

alc2:mar2 2.98601 0.46468 6.426 1.31e-10 \*\*\*

cig2:mar2 2.84789 0.16384 17.382 < 2e-16 \*\*\*

---

Null deviance: 2851.46098 on 7 degrees of freedom

Residual deviance: 0.37399 on 1 degrees of freedom

AIC: 63.417

Number of Fisher Scoring iterations: 4

> pearson <- summary.lm(homo.assoc)$residuals # Pearson residuals

> sum(pearson^2) # Pearson goodness-of-fit statistic

[1] 0.4011006

> leverage <- lm.influence(homo.assoc)$hat # leverage values

> std.resid <- pearson/sqrt(1 - leverage) # standardized residuals

> expected <- fitted(homo.assoc) # estimated expected frequencies

> cbind(count, expected, pearson, std.resid)

count expected pearson std.resid

1 911 910.383170 0.02044342 0.6333249

2 538 538.616830 -0.02657821 -0.6333249

3 44 44.616830 -0.09234564 -0.6333249

4 456 455.383170 0.02890528 0.6333249

5 3 3.616830 -0.32434090 -0.6333251

6 43 42.383170 0.09474777 0.6333249

7 2 1.383170 0.52447895 0.6333251

8 279 279.616830 -0.03688791 -0.6333249

Association Models:

> sexdata <- read.table("sex.dat", header=TRUE)

> attach(sexdata)

> uv <- premar\*birth

> premar <- factor(premar); birth <- factor(birth)

> LL.fit <- glm(count ~ premar + birth + uv, family=poisson)

> summary(LL.fit)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 4.10684 0.08951 45.881 < 2e-16 \*\*\*

premar2 -1.64596 0.13473 -12.216 < 2e-16 \*\*\*

premar3 -1.77002 0.16464 -10.751 < 2e-16 \*\*\*

premar4 -1.75369 0.23432 -7.484 7.20e-14 \*\*\*

birth2 -0.46411 0.11952 -3.883 0.000103 \*\*\*

birth3 -0.72452 0.16201 -4.472 7.74e-06 \*\*\*

birth4 -1.87966 0.24910 -7.546 4.50e-14 \*\*\*

uv 0.28584 0.02824 10.122 < 2e-16 \*\*\*

Null deviance: 431.078 on 15 degrees of freedom

Residual deviance: 11.534 on 8 degrees of freedom

AIC: 118.21

Number of Fisher Scoring iterations: 4

> u <- c(1,1,1,1,2,2,2,2,4,4,4,4,5,5,5,5)

> v <- c(1,2,4,5,1,2,4,5,1,2,4,5,1,2,4,5)

> row.fit <- glm(count ~ premar + birth + u:birth, family=poisson)

> summary(row.fit)

Coefficients: (1 not defined because of singularities)

Estimate Std. Error z value Pr(>|z|)

(Intercept) 4.98722 0.14624 34.102 < 2e-16 \*\*\*

premar2 -0.65772 0.13124 -5.011 5.40e-07 \*\*\*

premar3 0.46664 0.16266 2.869 0.004120 \*\*

premar4 1.50195 0.17952 8.366 < 2e-16 \*\*\*

birth2 -0.31939 0.19821 -1.611 0.107103

birth3 -0.72688 0.20016 -3.632 0.000282 \*\*\*

birth4 -1.49032 0.23745 -6.276 3.47e-10 \*\*\*

birth1:u -0.59533 0.06555 -9.082 < 2e-16 \*\*\*

birth2:u -0.40543 0.06068 -6.681 2.37e-11 \*\*\*

birth3:u -0.12975 0.05634 -2.303 0.021276 \*

birth4:u NA NA NA NA

Null deviance: 431.078 on 15 degrees of freedom

Residual deviance: 8.263 on 6 degrees of freedom

AIC: 118.94

Number of Fisher Scoring iterations: 4

> column.fit <- glm(count ~ premar + birth + premar:v, family=poisson)

> summary(column.fit)

Coefficients: (1 not defined because of singularities)

Estimate Std. Error z value Pr(>|z|)

Estimate Std. Error z value Pr(>|z|)

(Intercept) 1.40792 0.26947 5.225 1.74e-07 \*\*\*

premar2 -0.68466 0.29053 -2.357 0.018444 \*

premar3 0.78235 0.22246 3.517 0.000437 \*\*\*

premar4 2.11167 0.18958 11.138 < 2e-16 \*\*\*

birth2 0.54590 0.11723 4.656 3.22e-06 \*\*\*

birth3 1.59262 0.14787 10.770 < 2e-16 \*\*\*

birth4 1.51018 0.16420 9.197 < 2e-16 \*\*\*

premar1:v 0.58454 0.05930 9.858 < 2e-16 \*\*\*

premar2:v 0.49554 0.07990 6.202 5.57e-10 \*\*\*

premar3:v 0.20315 0.06538 3.107 0.001890 \*\*

premar4:v NA NA NA NA

Null deviance: 431.0781 on 15 degrees of freedom

Residual deviance: 7.5861 on 6 degrees of freedom

AIC: 118.26

Number of Fisher Scoring iterations: 4

----------------------------------------------------------------

Models for Matched Pairs

ratings <- matrix(c(175, 16, 54, 188), ncol=2, byrow=TRUE,

+ dimnames = list("2004 Election" = c("Democrat", "Republican"),

+ "2008 Election" = c("Democrat", "Republican")))

> mcnemar.test(ratings, correct=FALSE)

Clustered Categorical Responses: Marginal Models:

> abortion

gender response question case

1 1 1 1 1

2 1 1 2 1

3 1 1 3 1

4 1 1 1 2

5 1 1 2 2

6 1 1 3 2

7 1 1 1 3

8 1 1 2 3

9 1 1 3 3

...

5545 0 0 1 1849

5546 0 0 2 1849

5547 0 0 3 1849

5548 0 0 1 1850

5549 0 0 2 1850

5550 0 0 3 1850

> z1 <- ifelse(abortion$question==1,1,0)

> z2 <- ifelse(abortion$question==2,1,0)

> z3 <- ifelse(abortion$question==3,1,0)

> library(gee)

> fit.gee <- gee(response ~ gender + z1 + z2, id=case, family=binomial,

+ corstr="exchangeable", data=abortion)

> summary(fit.gee)

GEE: GENERALIZED LINEAR MODELS FOR DEPENDENT DATA

gee S-function, version 4.13 modified 98/01/27 (1998)

Model:

Link

Variance to Mean Relation: Binomial

Correlation Structure: Exchangeable

Coefficients:

Estimate Naive S.E. Naive z Robust S.E. Robust z

(Intercept) -0.125325730 0.06782579 -1.84775925 0.06758212 -1.85442135

gender 0.003437873 0.08790630 0.03910838 0.08784072 0.03913758

z1 0.149347107 0.02814374 5.30658404 0.02973865 5.02198729

z2 0.052017986 0.02815145 1.84779075 0.02704703 1.92324179

Working Correlation

[,1] [,2] [,3]

[1,] 1.0000000 0.8173308 0.8173308

[2,] 0.8173308 1.0000000 0.8173308

[3,] 0.8173308 0.8173308 1.0000000

> fit.gee2 <- gee(response ~ gender + z1 + z2, id=case, family=binomial,

+ corstr="independence", data=abortion)

> summary(fit.gee2)

Link: Logit

Variance to Mean Relation: Binomial

Correlation Structure: Independent

Coefficients:

Estimate Naive S.E. Naive z Robust S.E. Robust z

(Intercept) -0.125407576 0.05562131 -2.25466795 0.06758236 -1.85562596

gender 0.003582051 0.05415761 0.06614123 0.08784012 0.04077921

z1 0.149347113 0.06584875 2.26803253 0.02973865 5.02198759

z2 0.052017989 0.06586692 0.78974374 0.02704704 1.92324166

Working Correlation

[,1] [,2] [,3]

[1,] 1 0 0

: Logit

[2,] 0 1 0

[3,] 0 0 1

> geeglm(y ~ x1 + x2, family=binomial, id=subject, corst=’’exchangeable’’)

> insomnia<-read.table("insomnia.dat",header=TRUE)

> insomnia<-as.data.frame(insomnia)

> insomnia

case treat occasion outcome

1 1 0 1

1 1 1 1

2 1 0 1

2 1 1 1

3 1 0 1

3 1 1 1

4 1 0 1

4 1 1 1

5 1 0 1

...

239 0 0 4

239 0 1 4

> library(repolr)

> fit <- repolr(formula = outcome ~ treat + occasion + treat \* occasion,

+ subjects="case", data=insomnia, times=c(1,2), categories=4,

corstr = "independence")

> summary(fit$gee)

Coefficients:

Estimate Naive S.E. Naive z Robust S.E. Robust z

factor(cuts)1 -2.26708899 0.2027367 -11.1824294 0.2187606 -10.3633343

factor(cuts)2 -0.95146176 0.1784822 -5.3308499 0.1809172 -5.2591017

factor(cuts)3 0.35173977 0.1726860 2.0368745 0.1784232 1.9713794

treat 0.03361002 0.2368973 0.1418759 0.2384374 0.1409595

occasion 1.03807641 0.2375992 4.3690229 0.1675855 6.1943093

treat:occasion 0.70775891 0.3341759 2.1179234 0.2435197 2.9063728

Clustered Categorical Responses: Random Effects Models:

> abortion

gender response question case

1 1 1 1 1

2 1 1 2 1

3 1 1 3 1

4 1 1 1 2

5 1 1 2 2

6 1 1 3 2

...

5548 0 0 1 1850

5549 0 0 2 1850

5550 0 0 3 1850

> z1 <- ifelse(abortion$question==1,1,0)

> z2 <- ifelse(abortion$question==2,1,0)

> z3 <- ifelse(abortion$question==3,1,0)

> library(glmmML)

> fit.glmmML <- glmmML(response ~ gender + z1 + z2,

+ cluster=abortion$case, family=binomial, data=abortion,

+ method = ’ghq’, n.points=50, start.sigma=9)

> summary(fit.glmmML)

Call: glmmML(formula = response ~ gender + z1 + z2, family = binomial,

data = abortion, cluster = abortion$case, start.sigma = 9,

method = "ghq", n.points = 50)

coef se(coef) z Pr(>|z|)

(Intercept) -0.62222 0.3811 -1.63253 1.03e-01

gender 0.01272 0.4936 0.02578 9.79e-01

z1 0.83587 0.1599 5.22649 1.73e-07

z2 0.29290 0.1568 1.86822 6.17e-02

Scale parameter in mixing distribution: 8.788 gaussian

Std. Error: 0.5282

LR p-value for H\_0: sigma = 0: 0

Residual deviance: 4578 on 5545 degrees of freedom AIC: 4588

Beta-Binomial and Quasi-likelihood Analysis

> group <- rats$group

> library(VGAM) # We use Thomas Yee’s VGAM library

> fit.bb <- vglm(cbind(y,n-y) ~ group, betabinomial(zero=2,irho=.2),

data=rats)

# two parameters, mu and rho, and zero=2 specifies 0 covariates for 2nd

# parameter (rho); irho is the initial guess for rho in beta-bin variance.

> summary(fit.bb) # fit of beta-binomial model

Coefficients:

Value Std. Error t value

(Intercept):1 1.3458 0.24412 5.5130

(Intercept):2 -1.1458 0.32408 -3.5355 # This is logit(rho)

group2 -3.1144 0.51818 -6.0103

group3 -3.8681 0.86285 -4.4830

group4 -3.9225 0.68351 -5.7387

Names of linear predictors: logit(mu), logit(rho)

Log-likelihood: -93.45675 on 111 degrees of freedom

> logit(-1.1458, inverse=T) # This is a function in VGAM

[1] 0.2412571 # The estimate of rho in beta-bin variance

> install.packages("aod") # another way to fit beta-binomial models

> library(aod)

> betabin(cbind(y,n-y) ~ group, random=~1,data=rats)

Beta-binomial model

------------------betabin(formula

= cbind(y, n - y) ~ group, random = ~1, data = rats)

Fixed-effect coefficients:

Estimate Std. Error z value Pr(> |z|)

(Intercept) 1.346e+00 2.481e-01 5.425e+00 5.799e-08

group2 -3.115e+00 5.020e-01 -6.205e+00 5.485e-10

group3 -3.869e+00 8.088e-01 -4.784e+00 1.722e-06

group4 -3.924e+00 6.682e-01 -5.872e+00 4.293e-09

Overdispersion coefficients:

Estimate Std. Error z value Pr(> z)

phi.(Intercept) 2.412e-01 6.036e-02 3.996e+00 3.222e-05

> quasibin(cbind(y,n-y) ~ group, data=rats) # QL with beta-bin variance

Quasi-likelihood generalized linear model

-----------------------------------------

quasibin(formula = cbind(y, n - y) ~ group, data = rats)

Fixed-effect coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 1.2124 0.2233 5.4294 < 1e-4

group2 -3.3696 0.5626 -5.9893 < 1e-4

group3 -4.5853 1.3028 -3.5197 4e-04

group4 -4.2502 0.8484 -5.0097 < 1e-4

Overdispersion parameter:

phi

0.1923

Pearson’s chi-squared goodness-of-fit statistic = 54.0007

Non-Model Based Classification and Clustering

> library(tree)

> attach(crabs)

> fit <- rpart(y ~ color + width, method="class")

> plot(fit)

> text(fit)

> printcp(fit)

Classification tree:

rpart(formula = y ~ color + width, method = "class")

Variables actually used in tree construction:

[1] color width

Root node error: 62/173 = 0.35838

n= 173

CP nsplit rel error xerror xstd

1 0.161290 0 1.00000 1.00000 0.101728

2 0.080645 1 0.83871 1.03226 0.102421

3 0.064516 2 0.75806 0.96774 0.100972

4 0.048387 3 0.69355 0.93548 0.100149

5 0.016129 4 0.64516 0.85484 0.097794

6 0.010000 6 0.61290 0.82258 0.096728

> plotcp(fit)

> summary(fit)

> plot(fit, uniform=TRUE,

main="Classification Tree for Crabs")

> pfit2 <- prune(fit, cp= 0.02)

> plot(pfit2, uniform=TRUE,

main="Pruned Classification Tree for Crabs")

plot(pfit2, uniform=TRUE,

+ main="Pruned Classification Tree for Crabs")

> text(pfit2, use.n=TRUE, all=TRUE, cex=.8)

> post(pfit2, file = "ptree2.ps",

title = "Pruned Classification Tree for Crabs")

post(pfit2, file = "ptree2.ps",

+ title = "Pruned Classification Tree for Crabs")

Cluster Analysis

> x <- read.table("election.dat", header=F)

> x

V1 V2 V3 V4 V5 V6 V7 V8 V9

1 0 0 0 0 1 0 0 0

2 0 0 0 1 1 1 1 1

3 0 0 0 1 0 0 0 1

4 0 0 0 0 1 0 0 1

5 0 0 0 1 1 1 1 1

6 0 0 1 1 1 1 1 1

7 1 1 1 1 1 1 1 1

8 0 0 0 1 1 0 0 0

9 0 0 0 1 1 1 0 1

10 0 0 1 1 1 1 1 1

11 0 0 0 1 1 0 0 1

12 0 0 0 0 0 0 0 0

13 0 0 0 0 0 0 0 1

14 0 0 0 0 0 0 0 0

> distances <- dist(x,method="manhattan")

> states <- c("AZ", "CA", "CO", "FL", "IL", "MA", "MN",

"MO", "NM", "NY", "OH", "TX", "VA", "WY")

> democlust <- hclust(distances,"average")

> postscript(file="dendrogram-election.ps")

> plot(democlust, labels=states)

> graphics.off()

Read Excel file:

library(xlsx);

mydata<-read.xlsx("C:/Users/MHE/Desktop/ActiveCourses/Projects/BehavioralPricing/Data/cleaned10232013.xlsx",1)

Repeat columns:

matrix(rep(x,each=n), ncol=n, byrow=**TRUE**)

Pasted from <<http://www.r-bloggers.com/a-quick-way-to-do-row-repeat-and-col-repeat-rep-row-rep-col/>>

Repeat Rows

matrix(rep(x,each=n),nrow=n)

Pasted from <<http://www.r-bloggers.com/a-quick-way-to-do-row-repeat-and-col-repeat-rep-row-rep-col/>>

Define identity matrix:

Diag(n)

Create a Diagonal matrix with diagonal of a vector:

diag(5)\*c(1,2,3,4,5)

**Matrix facilites**

In the following examples, **A** and **B** are matrices and**x** and **b**are a vectors.

|  |  |
| --- | --- |
| **Operator or Function** | **Description** |
| **A \* B** | Element-wise multiplication |
| **A %\*% B** | Matrix multiplication |
| **A %o% B** | Outer product.**AB'** |
| **crossprod(A,B)**  **crossprod(A)** | **A'B** and **A'A**respectively. |
| **t(A)** | Transpose |
| **diag(x)** | Creates diagonal matrix with elements of **x** in the principal diagonal |
| **diag(A)** | Returns a vector containing the elements of the principal diagonal |
| **diag(k)** | If k is a scalar, this creates a k x k identity matrix. Go figure. |
| **solve(A, b)** | Returns vector **x** in the equation **b = Ax**(i.e., **A-1b**) |
| **solve(A)** | Inverse of **A** where A is a square matrix. |
| **ginv(A)** | Moore-Penrose Generalized Inverse of **A**.  ginv(A) requires loading the**MASS** package. |
| **y<-eigen(A)** | **y$val**are the eigenvalues of **A**  **y$vec** are the eigenvectors of **A** |
| **y<-svd(A)** | Single value decomposition of**A**.  **y$d** = vector containing the singular values of **A**  **y$u** = matrix with columns contain the left singular vectors of **A**  **y$v**= matrix with columns contain the right singular vectors of **A** |
| **R <- chol(A)** | Choleski factorization of **A**. Returns the upper triangular factor, such that **R'R = A**. |
| **y <- qr(A)** | QR decomposition of **A**.  **y$qr**has an upper triangle that contains the decomposition and a lower triangle that contains information on the Q decomposition.  **y$rank** is the rank of A.  **y$qraux** a vector which contains additional information on Q.  **y$pivot** contains information on the pivoting strategy used. |
| **cbind(A,B,...)** | Combine matrices(vectors) horizontally. Returns a matrix. |
| **rbind(A,B,...)** | Combine matrices(vectors) vertically. Returns a matrix. |
| **rowMeans(A)** | Returns vector of row means. |
| **rowSums(A)** | Returns vector of row sums. |
| **colMeans(A)** | Returns vector of column means. |
| **colSums(A)** | Returns vector of coumn means. |

Pasted from <<http://www.statmethods.net/advstats/matrix.html>>

Convert matrix to vector:

As.vector(A)

<http://stackoverflow.com/questions/3823211/how-to-convert-a-matrix-to-a-1-dimensional-array-in-r>

Text Mining Using R:

# preprocessing of the document

library(tm)

firefox.csv<-read.csv("c:/CDBookSurvay/Comments.csv")

firefox <- Corpus(DataframeSource(firefox.csv)) # create corpus for analysis

firefoxcopy <- firefox # keep a copy of corpus to use later as a dictionary for stem completion

firefox <-tm\_map(firefox, tolower) # convert to lower case

firefox <- tm\_map(firefox, removeNumbers) # remove numbers

for (j in 1:length(firefox)) firefox[[j]] <- gsub("'", " ",firefox[[j]])# to remove special puncutation but not connect

firefox <- tm\_map(firefox, removePunctuation)# remove punctuations

firefox <- tm\_map(firefox, removeWords, stopwords("english")) #remove stop words

newstopwords <- c("and", "for", "the", "to", "in", "when", "then", "he", "she", "than", "a", "for", "it", "of", "on", "to","im")

firefox <- tm\_map(firefox, removeWords, newstopwords)

firefox <- tm\_map(firefox, stemDocument)# stem words

inspect(firefox[1:10])

firefox <- tm\_map(firefox, stemCompletion, dictionary=firefoxcopy) #stem completion

inspect(firefoxcopy[1:10])

summary(firefox)

myTdm <- TermDocumentMatrix(firefox, control = list(wordLengths=c(1,Inf)))

myTdm # printing dtm summery

idx <- which(dimnames(myTdm)$Terms =="alexa")

inspect(myTdm[idx+(0:5),1:10]) # look at 5 keywords after the keyword alexa over 10 documents that used for dtm

inspect(myTdm[0:20,1:10]) # check items of dtm

rownames(myTdm) # write all the keywords you have used

findFreqTerms(myTdm, lowfreq=3) #find frequent terms

# plot of more frequent words

termFrequency <- rowSums(as.matrix(myTdm)) # go over matrix and filtering for drawing a plot

termFrequency <- subset(termFrequency, termFrequency>=3) # go for terms that are in text more than 3 times

library(ggplot2) # use graphic package to draw plots

qplot(names(termFrequency), termFrequency, geom="bar") + coord\_flip() # draw horizontal bar plot

barplot(termFrequency, las=2) # draw vertical bar plot

findAssocs(myTdm, "love", 0.25)# find words with highest asociation

library(wordcloud) # used for importance of the word check

m <- as.matrix(myTdm) # convert document term matrix to normal matrix

# calculate the frequency of words and sort it descendingly by frequency

wordFreq <- sort(rowSums(m), decreasing=TRUE)

# word cloud

set.seed(375) # to make it reproducible

grayLevels <- gray( (wordFreq+10) / (max(wordFreq)+10) )

# frequency below 1 is not ploted in the following

# random.order=F: frequent words plotted first in the center of the cloud

# set colour to: grayLevels or raingbow() to colorful or gray map

wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=2, random.order=F,colors=grayLevels)

# clustering

# remove sparse terms

# you can remove sparce terms to avoid being flooded with words

myTdm2 <- removeSparseTerms(myTdm, sparse=0.95)

m2 <- as.matrix(myTdm2)

# cluster of terms/words (come together e.g. couple of twits on text mining analysis, and couple of twits on job vacancies in PhD in different clusters)

distMatrix <- dist(scale(m2)) # calculate distance between terms after scaling

fit <- hclust(distMatrix, method="ward") # clustering agglomeration method is set to ward: icreased variance when two clusters are merged; other options are: single linkage, complete linkage, average linkage, median and centroid

plot(fit)

# cut tree into 10 clusters

rect.hclust(fit, k=10) # cut into 10 clusters

(groups <- cutree(fit, k=10))

# clustering using k-min of documents

# transpose the matrix to cluster documents (tweets)

m3 <- t(m2) # take value of matrix as numeric & transpose to document term

# set a fixed random seed

set.seed(122) # to produce the clustering result

# k-means clustering of tweets

k <- 3 # 8 clusters

kmeansResult <- kmeans(m3, k)

# cluster centers

round(kmeansResult$centers, digits=3) # popular words in cluster and center

# check k mean cluster by top 3 words

for (i in 1:k) {

cat(paste("cluster ", i, ": ", sep=""))

s <- sort(kmeansResult$centers[i,], decreasing=T)

cat(names(s)[1:3], "\n")

# print the tweets of every cluster

# print(rdmTweets[which(kmeansResult$cluster==i)])

}

library(Rgraphviz)# to use for cluster assowciation matrix

plot(myTdm, terms = findFreqTerms(myTdm, lowfreq = 1)[1:20], corThreshold = 0)

library(fpc)#draw cluster based on matrix

plotcluster(m3, kmeansResult$cluster)

library(fpc) # clustering with Partitioning Around Medoids (PAM): (representative objects) more robust to noise and outliers than k-means clustering

# partitioning around medoids with estimation of number of clusters

pamResult <- pamk(m3, metric = "manhattan") # estimate number of optimal clusters

# number of clusters identified

(k <- pamResult$nc)

pamResult <- pamResult$pamobject

# print cluster medoids

for (i in 1:k) {

cat(paste("cluster", i, ": "))

cat(colnames(pamResult$medoids)[which(pamResult$medoids[i,]==1)], "\n")

#print tweets in cluster i

# print(rdmTweets[pamResult$clustering==i])

}

# plot clustering result

layout(matrix(c(1,2),2,1)) # set to two graphs per page

plot(pamResult, color=F, labels=4, lines=0, cex=.8, col.clus=1,

col.p=pamResult$clustering)

layout(matrix(1)) # change back to one graph per page

#create social network of terms

termDocMatrix<-m2

termDocMatrix[1:5,1:5] # check Tdm

# change it to a Boolean matrix

termDocMatrix[termDocMatrix>=1] <- 1

# transform into a term-term adjacency matrix

termMatrix <- termDocMatrix %\*% t(termDocMatrix) # %\*% product of two matrices

# inspect terms numbered 5 to 7

termMatrix[5:7,5:7]

library(igraph)

# build a graph from the above matrix

g <- graph.adjacency(termMatrix, weighted=T, mode = "undirected")

# remove loops

g <- simplify(g)

# set labels and degrees of vertices

V(g)$label <- V(g)$name

V(g)$degree <- degree(g)

# set seed to make the layout reproducible

set.seed(3952)

layout1 <- layout.fruchterman.reingold(g)

plot(g, layout=layout1)

#dynamically rearranged layout get detail by running ?igraph::layout

plot(g, layout=layout.kamada.kawai)

tkplot(g, layout=layout.kamada.kawai)#extremely interesting graph creation

pdf("term-network.pdf") # put terms plot in a pdf file

plot(g, layout=layout.fruchterman.reingold)

dev.off()

# size of plot's term according to the degree: important terms stand out

# set the width and transparency of edges based on their weights

# vertices and edges are accessed with V() and E()

# rgb(red, green, blue,alpha) defines a color with an alpha transparency

V(g)$label.cex <- 2.2 \* V(g)$degree / max(V(g)$degree)+ .2

V(g)$label.color <- rgb(0, 0, .2, .8)

V(g)$frame.color <- NA

egam <- (log(E(g)$weight)+.4) / max(log(E(g)$weight)+.4)

E(g)$color <- rgb(.5, .5, 0, egam)

E(g)$width <- egam

# plot the graph in layout1

plot(g, layout=layout1)

#build network of documents (tweets) first phase

# remove "r", "data" and "mining" most used if they make the document crowded

# idx <- which(dimnames(termDocMatrix)$Terms %in% c("r", "data", "mining"))

#M <- termDocMatrix[-idx,] # remove terms from matrix

M<-termDocMatrix # since I did not wanted to remove anything

# build a tweet-tweet adjacency matrix

tweetMatrix <- t(M) %\*% M

library(igraph)

g <- graph.adjacency(tweetMatrix, weighted=T, mode = "undirected")

V(g)$degree <- degree(g)

g <- simplify(g)

# set labels of vertices to tweet IDs

V(g)$label <- V(g)$name

V(g)$label.cex <- 1

V(g)$label.color <- rgb(.4, 0, 0, .7)

V(g)$size <- 2

V(g)$frame.color <- NA

barplot(table(V(g)$degree)) # check degree distribution of vertices

#build network of documents (tweets) second phase

idx <- V(g)$degree == 0

V(g)$label.color[idx] <- rgb(0, 0, .3, .7) # set based on degree

# set labels to the IDs and the first 10 characters of tweets

# limit to the first 20 character of every tweet

# label of each set to tweet ID so that graph would not be overcrowded

# set color and width of edges based on their weights

#V(g)$label[idx] <- paste(V(g)$name[idx], substr(df$text[idx], 1, 20), sep=" ")

egam <- (log(E(g)$weight)+.2) / max(log(E(g)$weight)+.2)

E(g)$color <- rgb(.5, .5, 0, egam)

E(g)$width <- egam

set.seed(3152)

layout2 <- layout.fruchterman.reingold(g)

plot(g, layout=layout2)

# remove isolated vertices and draw again

g2 <- delete.vertices(g, V(g)[degree(g)==0])

plot(g, layout=layout2)

# remove edges with low degree and draw again

g3 <- delete.edges(g, E(g)[E(g)$weight <= 1])

g3 <- delete.vertices(g3, V(g3)[degree(g3) == 0])

plot(g3, layout=layout.fruchterman.reingold)

# look at specific clique: considerably connected {replacement for dftext

inspect(firefox[c(15,16)])

#graph g directly from termDocMatrix

# create a graph

g <- graph.incidence(termDocMatrix, mode=c("all"))

# get index for term vertices and tweet vertices

nTerms <- nrow(M)

nDocs <- ncol(M)

idx.terms <- 1:nTerms

idx.docs <- (nTerms+1):(nTerms+nDocs)

# set colors and sizes for vertices

V(g)$degree <- degree(g)

V(g)$color[idx.terms] <- rgb(0, 1, 0, .5)

V(g)$size[idx.terms] <- 6

V(g)$color[idx.docs] <- rgb(1, 0, 0, .4)

V(g)$size[idx.docs] <- 4

V(g)$frame.color <- NA

# set vertex labels and their colors and sizes

V(g)$label <- V(g)$name

V(g)$label.color <- rgb(0, 0, 0, 0.5)

V(g)$label.cex <- 1.4\*V(g)$degree/max(V(g)$degree) + 1

# set edge width and color

E(g)$width <- .3

E(g)$color <- rgb(.5, .5, 0, .3)

set.seed(958)#5365, 227

plot(g, layout=layout.fruchterman.reingold)

# returns all vertices of "love" # if node does not exist returns "invalid vertex name"

V(g)[nei("love")]

V(g)[neighborhood(g, order=1, "love")[[1]]]# alternative way of geting vertices

#check which vertices include all three elements "thank", "perfect", "love"

(rdmVertices <- V(g)[nei("love") & nei("perfect") & nei("thank")])

inspect(firefox[as.numeric(rdmVertices$label)])# check content of the doc that includes these three terms

# remove three words to see the relationship with doc with other words

idx <- which(V(g)$name %in% c("love", "perfect", "thank"))

g2 <- delete.vertices(g, V(g)[idx-1])

g2 <- delete.vertices(g2, V(g2)[degree(g2)==0])

set.seed(209)

plot(g2, layout=layout.fruchterman.reingold)

Global Variable in R:

Variables declared inside a function are local to that function. For instance:

foo <- function() {  
 bar <- 1  
}  
foo()  
bar

gives the following error: Error: object 'bar' not found.

If you want to make bar a global variable, you should do:

foo <- function() {  
 bar <<- 1  
}  
foo()  
bar

In this case bar is accessible from outside the function.

However, unlike C, C++ or many other languages, brackets do not determine the scope of variables. For instance, in the following code snippet:

if (x > 10) {  
 y <- 0  
}  
else {  
 y <- 1  
}

y remains accessible after the if-else statement.

As you well say, you can also create nested environments. You can have a look at these two links for understanding how to use them:

Pasted from <<http://stackoverflow.com/questions/10904124/global-and-local-variables-in-r>>

To access a global variable in R you do not need to do anything you just use the name.

For example, from ?Sys.sleep

testit <- function(x)  
{  
 p1 <- proc.time()  
 Sys.sleep(x)  
 proc.time() - p1 # The cpu usage should be negligible  
}  
testit(3.7)

Yielding

> testit(3.7)  
 user system elapsed   
 0.000 0.000 3.704

Pasted from <<http://stackoverflow.com/questions/1174799/how-to-make-execution-pause-sleep-wait-for-x-seconds>>

Ways to pause the program:

 'par(ask = TRUE)'

Pasted from <<http://tolstoy.newcastle.edu.au/R/help/04/11/8084.html>>

Readline()

Elementwise comparison of two vectors:

Assuming that the vectors x and y are of the same length, pmax is your function.

z = pmax(x, y)

If the lengths differ, the pmax expression will return different values than your loop, due to recycling.

Pasted from <<http://stackoverflow.com/questions/14092922/finding-maximum-of-two-vectors-without-a-loop>>

Break Program while executing keyboard shortcut: ESC

With Heterogeneity Model for Behavioral pricing (Regret Project):

rm(list = ls());

# model with heterogeneity without fixed effect on the data

#use Gradient Methods, Genetic Algorithim, and ...

# parameters to estimate are: [alpha c bp alphap betar] where alpha is

# not fixed effect here, but an intercept

# alpha\_e c\_e bp\_e alphap\_e betar\_e\*c

#defining functions

# function to conduct contraction mapping and over real data so include

# Durations as well, and this is for cost heterogeneity

FuncWithHetroWithRegrtRD = function(x){

#global P1 P2 Dur1 Dur2 lambda gamma shares km cost; #outside\_n

#global vcm se\_est betas variance

#gamma: the discount factor

#P1: price for first period

#P2: price for 2nd period

#lambda: Availability of second period

#arranging matrixes

#J: number of products under study = 106 in hour example

#T: number of periods =2 in hour example

#heterogeneous beta includes beta\_ip, beta\_ir, alpha\_ip

J = dim(shares)[1];

T = dim(shares)[2];

# parameters of heterogeneity [pi1 bp alphap betar\*c]

pi1 = exp(x[1])/(1+exp(x[1])); #share of first segment (use transformation to make sure it is between zero and one)

# use transformation to make sure that it is lower

bp = -exp(x[2]); #price coefficient difference heterogeneity coeff

alphap = -exp(x[3]); #high price regret difference heterogeneity coeff

betar = -exp(x[4]); #stock out regret difference heterogeneity coeff

# cat('input parameters for function are for [pi1 bp alphap betar] are: \n')

# cbind(pi1,bp,alphap,betar);

# pause

dd = matrix(rep(0,(J\*T)),nrow=J,ncol=T); #since I have three periods and

uij1 = matrix(rep(0,J),nrow=J,ncol=2);

uij2 = matrix(rep(0,J),nrow=J,ncol=2);

k = 100;

de1 = dd;

i = 0; # track contraction mapping

#contraction mapping

cat(k,'k','\n')

cat(km,'km','\n')

while(k>km){

i = i+1;

if (ceiling(i/1000) == (i/1000)){

cat(i,'\n');

#median(as.vector(de1)-as.vector(de))

        if (ceiling(i/1000)>80){

                stop("too many iterations");

        }

}

de = de1;

# calculate utility

uij1[,1] = de[,1];

uij2[,1] = de[,2];

uij1[,2] = de[,1]+bp\*P1+ alphap\*lambda\*(P1-P2);

uij2[,2] = de[,2]+gamma\*(lambda\*(bp\*P2)+ betar\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2));

e1=exp(uij1); e2=exp(uij2); #\*cost

shares\_e=cbind((e1/(1+e1+e2))%\*%matrix(c(pi1,1-pi1),nrow=2,ncol=1),(e2/(1+e1+e2))%\*%matrix(c(pi1,1-pi1),nrow=2,ncol=1));

shares\_e=pmax(shares\_e,0.00000001); #As a precaution

de1 = de + log(shares)- log(shares\_e) ;

k = max(abs(as.vector(de1)-as.vector(de)));

#cat(k,'\n');

}

dd = de1; # first segment utility portion

# run regression to find linear parameters

shares\_n = matrix(t(dd),nrow=J\*2,ncol=1); #stack shares on top of eachother

#cat(shares\_n)

#pause

# with fixed effect models

# p=J+4;

# X1= cbind(diag(J),(0.5\*Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(diag(J)\*gamma\*lambda,0.5\*gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2));

# heterogeneity in consumption utility explained by cost

p = 5;

X1 = cbind(rep(1,J),(Dur1+gamma\*Dur2)\*cost,P1,lambda\*(P1-P2),rep(0,J));

X2= cbind(gamma\*lambda,gamma\*lambda\*Dur2\*cost,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(Dur1+gamma\*Dur2)\*cost);

X = t(cbind(X1,X2));

Xn= matrix(X,ncol=J\*2, nrow=p);

Xn=t(Xn); #stack X's

Yn= shares\_n;

#log(shares\_n./outside\_n);

#OLS global setting

betas<<-solve(t(Xn)%\*%Xn)%\*%t(Xn)%\*%Yn;

errors<<-Yn-Xn%\*%betas;

vcm<<-as.vector(t(errors)%\*%errors)\*solve(t(Xn)%\*%Xn)/(2\*J);

se\_est<<-sqrt(diag(vcm));

#OLS Local setting

betas=solve(t(Xn)%\*%Xn)%\*%t(Xn)%\*%Yn;

errors=Yn-Xn%\*%betas;

vcm=as.vector(t(errors)%\*%errors)\*solve(t(Xn)%\*%Xn)/(2\*J);

se\_est=sqrt(diag(vcm));

# calculate variance

variance <<- colMeans(errors^2)\*2\*J/(2\*J-1);

# to avoid Jacobian that is zero, which will create Log(0)= NaN; + ones(size(esh1,1)

s1 = e1/(1+e1+e2);

s11 = 1-s1;

s1=pmax(s1,0.00000001); #As a precaution

s11=pmax(s11,0.00000001); #As a precaution

s2 = e2/(1+e1+e2);

s21 = 1-s2;

s2=pmax(s2,0.00000001); #As a precaution

s21=pmax(s21,0.00000001); #As a precaution

Jacobian = cbind((s1\*s11)%\*%matrix(c(pi1,1-pi1)),(s2\*s21)%\*%matrix(c(pi1,1-pi1)));

LogJacobian = -sum(log(as.vector(Jacobian)));

LogDemandShockLikelihood = - T\*J\*log(sqrt(2\*pi\*variance)) - .5\*colSums(errors^2/variance);

likelihood = LogDemandShockLikelihood + LogJacobian;

y = - likelihood ;

#cat ('set of (Jacobian, likelihood, Log demand shock Likelihood) is:\n');

#cat (cbind(LogJacobian,likelihood,LogDemandShockLikelihood),'\n');

#readline()

return(y);

}

#data

#global P1 P2 Dur1 Dur2 lambda gamma shares km cost #outside\_n

#global vcm se\_est betas variance

library(xlsx);

num<-read.xlsx("C:/Users/MHE/Desktop/ActiveCourses/Projects/BehavioralPricing/Data/cleaned10232013.xlsx",1)

#cbind(summary(num))

names(num)

J = dim(num)[1];

p = dim(num)[2];

Dur1=num[,2];

Dur2=num[,3];

P1=num[,4];

P2=num[,5];

Av1=num[,6];

Av2=num[,7];

Av3=num[,12];

S1=num[,8];

S2=num[,9];

MKTSz=num[,10];

cost = num[,11];

T = 3;

# test for cut off of second period

# P3=num[,13];

# P4=num[,16];

# Dur3=num[,14];

# S3=num[,15];

# S2=S3;

# P2=P4;

#Dur2=Dur3;

# test for average of second period availability, and normalized S2

lambda = Av2;

# use normalize availability

#lambda = Av2/Av1; #availability

#lambda = Av3; # availability at the beggining of second period

# test for normalized sales of second period

MKTSz=MKTSz/1.25;

cf=1.25;

MKTSz=MKTSz\*cf;

MKTSz = MKTSz - (cf\*S2);

S2 = S2/lambda;

MKTSz = MKTSz + (cf\*S2);

# tstat=regstats(P1,P1-P2,'linear'), tstat.rsquare=.711 => VIF=.711

#gamma =.975; #discount factor

gamma=1/(1+.0025)^Dur1;

# create shares

shares=cbind(S1/MKTSz,S2/MKTSz);

outside=matrix(rep(rep(1,J)-rowSums(shares),each=2), ncol=2, byrow=TRUE);

shares\_n=matrix(shares,nrow=J\*2); #stack shares on top of eachother

outside\_n=matrix(t(outside),nrow=J\*2); #stack outside share

# parameters of heterogeneity [pi1 bp alphap betar\*c] difference with main

# cat('main parameters for [pi1 bp alphap betar] are:\n')

#cbind(pi1,bpd,alphapd,betard\*c)

#readline()

km = 0.001;

shares=pmax(shares,0.00000001); #As a precaution

Rprof("AggregLogitWHeterogen.out")

#X0 = c(0.5,0.2,0.3,0.4);

#X0 = c(0.1,0.1,0.1,0.1);

X0= c(0.5,0.5,0.5,0.5);

#X0= matrix(c(0.5,0.5,0.5,0.8),nrow=1,ncol=4);

#X0= rep(0,4);

#X0= c(0.8,0.8,1,0.8);

#c(log(pi1/(1-pi1)),log(-bpd),log(-alphapd),log(-betard\*c));

#options=optimset('Display','on','MaxIter',10000,'TolX',10^-30,'TolFun',10^-30);

# no fixed effect

#[x,fval,exitflag,output,grad,hessian]=fminunc('FuncWithHetroWithRegrtRD',X0,options);

#fixed effect heterogeneity

#[x,fval,exitflag,output,grad,hessian]=fminunc('FuncWithHetroWithRegrtRDFE',X0,options);

# heterogeneity by cost

#first optimization method

out <- nlm(FuncWithHetroWithRegrtRD , X0, hessian = TRUE,

print.level = 2)

print(out)

fval=out$minimum;

x=out$estimate;

exitflag=out$code;

grad=out$gradient;

hessian=out$hessian;

#second optimization method

library(numDeriv)

out <- nlminb(X0, FuncWithHetroWithRegrtRD)

print(out);

x=out$par;

hessian <- hessian(func=FuncWithHetroWithRegrtRD, x=out$par)

#third Optimization Method

# method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent"

#method that worked well: L-BFGS-B

#Method "L-BFGS-B" is that of Byrd et. al. (1995) which allows box constraints, that is each variable can be given a lower and/or upper bound. The initial value must satisfy the constraints. This uses a limited-memory modification of the BFGS quasi-Newton method. If non-trivial bounds are supplied, this method will be selected, with a warning.

out <- optim(X0, FuncWithHetroWithRegrtRD, method="L-BFGS-B",

control = list(maxit = 30000, temp = 2000, trace = TRUE,

REPORT = 500), hessian=T) #, fnscale=-1

x = out$par;

hessian = out$hessian;

fval = out$value;

#Fourth optimization Method: Genetic Algorithm

library(rgenoud)

out <- genoud(FuncWithHetroWithRegrtRD, nvars=4,max=FALSE) #didn't work

library(DEoptim)

lower <- c(-10,-10)

upper <- -lower

## run DEoptim and set a seed first for replicability

#set.seed(1234)

#DEoptim(FuncWithHetroWithRegrtRD, lower, upper)

x = out$par;

hessian = out$hessian;

fval = out$value;

#[x,fval,exitflag,output,grad,hessian]=fminunc('FuncWithHetroWithRegrtRDCH',X0,options);

cat('estimation time is:\n');

Rprof(NULL);

#params=cbind(alpha,c,bp(1,1),alpha(1,1),betar(1,1));

# no fixed effect simple intercept

p = 5;

betas = t(betas);

se\_est = t(se\_est);

a\_e = t(betas[1,1:(p-4)]);

c\_e = betas[1,(p-3)];

bp\_e = betas[1,(p-2)];

alphap\_e= betas[1,(p-1)];

tt1\_e = betas[1,p];

betar\_e =tt1\_e/c\_e;

STEFOC=cbind(1/c\_e,-tt1\_e/(c\_e^2));

#ParamCovar =vcm[c(2,5),c(2,5)]\*J;

ParamCovar =vcm[c(p-3,p),c(p-3,p)]\*(2\*J);

betarSTE=sqrt(STEFOC%\*%ParamCovar%\*%t(STEFOC)/(2\*J));

cat('threshold parameter for contraction mapping is:\n');

cat(km,'\n');

#cat('Seed for random generation is:\n');

#cat(SEED1);

# c bp alphap betar

# cat('parameters estimation for: a c bp alphap betar are:\n');

# cat(cbind(t(betas[1,1:4]),betar\_e),'\n');

#cat(cbind(t(betas[1,1:4]/se\_est[1,1:4]),betar\_e/betarSTE]),'\n');

cat('estimation and t-stat for direct estimation for (c,bp,alphap,betar) is:\n');

cat(cbind(t(betas[1,(p-3):(p-1)]),betar\_e), '\n');

cat(cbind(t(se\_est[1,(p-3):(p-1)]),betarSTE),'\n');

cat(cbind(t(betas[1,(p-3):(p-1)]/se\_est[1,(p-3):(p-1)]),betar\_e/betarSTE),'\n');

# intercept, single intercept

cat('Intercept is: \n');

cat(betas[1,(p-4)],'\n');

cat(se\_est[1,(p-4)],'\n');

cat(betas[1,(p-4)]/se\_est[1,(p-4)],'\n');

# parameters of heterogeneity [pi betap alphap betar]

ste = diag(solve(hessian));

ste = sqrt(ste);

trat = cbind(exp(x[1])/(1+exp(x[1])),t(-exp(x[2:3])/ste[2:3]));

tth1\_e=-exp(x[4]);

betarh\_e =tth1\_e/c\_e;

STEFOCh=cbind(1/c\_e,-tth1\_e/(c\_e^2));

#ParamCovar =vcm[c(2,5),c(2,5)]\*2\*J;

ParamCovarh =diag(2)\*c(vcm[p-3,p-3],ste[4]^2)\*(2\*J);

betarSTEh=sqrt(STEFOCh%\*%ParamCovarh%\*%t(STEFOCh)/(2\*J));

cat('parm estimates for heterogeneity (pi,bp,alphap,betar) are:\n');

cat(cbind(exp(x[1])/(1+exp(x[1])),t(-exp(x[2:3])), betarh\_e),'\n');

cat(cbind(t(ste[1:3]),betarSTEh));

cat(cbind(t(trat[1:3]),betarh\_e/betarSTEh));

cat('log likelihood value is:\n');

cat(-fval);

LL=-fval;

AIC = 2\*p-2\*LL;

BIC = -2\*LL+p\*log(J);

cat('Log Likelihood, AIC, BIC is:\n');

cat(cbind(LL,AIC,BIC),'\n');

# regret coefficient

#[betar; betas(1,5)/(alpha+c+gamma\*c); betas(1,6)/betas(1,3);betarmean]

#se\_est';

GMM code of Regret pricing Project:

rm(list = ls());

# GMM Function of full model analysis

MeisamGMMfunc = function(p){

#global D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 C1 C2 P1 P2 J cost D11 r D12 MKTSz

# parameters: (bp,ah,a, c, tt)

bp = -exp(p[1]);

ah = -exp(p[2]);

a = exp(p[3]);

c = exp(p[4]);

tt = p[5];

v = p[6]^2; # to make sure that variance is positive

rho = exp(p[7])/(1+exp(p[7])); # assuming autocorrelation

#rp = exp(p[7]);

# F.O.C is summarized to:

# F.O.C is summarized to:

# 1. E(D1\*bp+D2\*ah+C1)=0

# 2. E(D3\*bp+D4\*ah+C2)=0

# 3. E(D5-a-D6\*c+bp\*P1+D7\*ah)=0

# 4. E(D8-D9\*a+D11\*c+r.\*bp\*P2+D10\*tt1)=0

# 5. E((D5-a-D6\*c+bp\*P1+D7\*ah)^2)-v=0

# 6. E((D8-D9\*a+D11\*c+r.\*bp\*P2+D10\*tt1)^2)-v=0

# 5. E((D5-a-D6\*c+bp\*P1+D7\*ah)\*(D8-D9\*a+D11\*c+r.\*bp\*P2+D10\*tt1))=0

#.\*cost

y1 = D1\*bp+C1+D2\*ah; #

y2 = D3\*bp+C2+D4\*ah; #

y3 = -D5+a+D6\*c\*cost+bp\*P1+D7\*ah; #

y4 = -D8+D9\*a+D11\*c\*cost+r\*bp\*P2+D10\*tt\*cost;#

y5 = (-D5+a+D6\*c\*cost+bp\*P1+D7\*ah)^2-v; #

y6 = (-D8+D9\*a+D11\*c\*cost+r\*bp\*P2+D10\*tt\*cost)^2-v; #

y7 = (-D5+a+D6\*c\*cost+bp\*P1+D7\*ah)\*(-D8+D9\*a+D11\*c\*cost+r\*bp\*P2+D10\*tt\*cost)-rho\*v;#

sig = cbind(y1,y2,y3,y4,y5,y6,y7);

cat(dim(sig),'sigma dim\n');

sig = (t(sig)%\*%sig)/J;

sig <<- (t(sig)%\*%sig)/J;

cat(dim(sig),'sigma dim\n');

psi = t(cbind(mean(y1),mean(y2),mean(y3),

mean(y4),mean(y5),mean(y6),

mean(y7)));

cat(dim(psi),'dim of psi \n')

cat(dim(ginv(sig)),'dim of siginv \n')

llh = t(psi)%\*%ginv(sig)%\*%psi\*J;

return (llh);

}

#global D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 C1 C2 P1 P2 J cost D11 r D12 MKTSz

library(xlsx);

num<-read.xlsx("C:/Users/MHE/Desktop/ActiveCourses/Projects/BehavioralPricing/Data/cleaned10232013.xlsx",1)

#cbind(summary(num))

names(num)

J = dim(num)[1];

p = dim(num)[2];

Dur1=num[,2];

Dur2=num[,3];

P1=num[,4];

P2=num[,5];

Av1=num[,6];

Av2=num[,7];

Av3=num[,12];

S1=num[,8];

S2=num[,9];

MKTSz=num[,10];

cost = num[,11];

T = 3;

A = Av2;

# normalized sales of second period

MKTSz=MKTSz/1.25;

cf=1.25;

MKTSz=MKTSz\*cf;

MKTSz = MKTSz - (cf\*S2);

S2 = S2/A;

MKTSz = MKTSz + (cf\*S2);

r=1./(1+.0025)^Dur1;

shares=cbind(S1/MKTSz,S2/MKTSz);

#normalize Market size

MKTSz=MKTSz/10000;

# put back shares so that it is used in calculation of F.O.C components

S1=shares[,1];

S2=shares[,2];

outside=matrix(rep(rep(1,J)-rowSums(shares),each=2), ncol=2, byrow=TRUE);

d=rep(1,J)-P2/P1; # (1-d)P1=P2

# F.O.C is summarized to:

# 1. E(D1\*bp+D2\*ah+C1)=0

# 2. E(D3\*bp+D4\*ah+C2)=0

# 3. E(D5-a-D6\*c+bp\*P1+D7\*ah)=0

# 4. E(D8-D9\*a+D11\*c+r.\*bp\*P2+D10\*tt1)=0

# 5. E((D5-a-D6\*c+bp\*P1+D7\*ah)^2)-v=0

# 6. E((D8-D9\*a+D11\*c+r.\*bp\*P2+D10\*tt1)^2)-v=0

# 5. E((D5-a-D6\*c+bp\*P1+D7\*ah)\*(D8-D9\*a+D11\*c+r.\*bp\*P2+D10\*tt1))=0

# tt1=br\*c

#parameters are: (bp,ah,a, c, tt1,v)

D1=S1\*(P1-cost)-S1^2\*(P1-cost)-r\*A\*(1-d)\*S1\*S2\*(P1-cost)+r^2\*A\*(1-d)^2\*S2\*(P1-cost)-S1\*S2\*(1-d)\*(P1-cost)\*r-r^2.\*A\*(1-d)^2.\*S2^2\*(P1-cost);

D2=P1\*d\*S1-A\*d\*S1^2\*P1-A\*d\*S1\*S2\*(1-d)\*P1^r;

C1=S1+r\*(1-d)\*S2;

D3=r\*A\*P1\*(P1-cost)\*S1\*S2-r^2\*A\*P1\*(P1-cost)\*(1-d)\*S2+r^2\*A\*P1\*(P1-cost)\*(1-d)\*S2^2;

D4=A\*P1^2\*S1\*P1-A\*S1^2\*P1^2-r\*A\*S1\*S2\*(1-d)\*P1^2;

C2=-r\*S2\*P1;

D5=shares[,1]-outside[,1];

D6=(0.5\*Dur1+r\*Dur2); #consider duration effect on consumption

D7=A\*(P1-P2);

D8=shares[,2]-outside[,2];

D9=A\*r;

D11 = A\*r\*Dur2\*.5; #include duration of usage into the model

D10=r\*(1-A)\*(.5\*Dur1+r\*Dur2); #include duration of usage in the model

D12 =r\*A\*(P1-P2);#.\*cost

# parameters: (bp,ah,a, c, tt, v, kt)

init\_p = c(-0.018,-0.1,0.1,0.6,-0.08,1,.5);

# c(-0.3,-3,.5,1,-0.8,20,.2);

# c(-0.018,-0.04,0.1,0.5,-0.018,10,.01);

# c(-2,-1,1,1,1,-1,1);

# c(-0.2,-0.3,0.3,0.5,0.1,1);

# rep(0,6);

# c(-0.01,-.3,0.1,0.5,0.1,1);

Rprof("GMM.out");

# heterogeneity by cost

#first optimization method

library(MASS)

out <- nlm(MeisamGMMfunc, init\_p, hessian = TRUE,

print.level = 2)

print(out);

fval=out$minimum;

param=out$estimate;

exitflag=out$code;

grad1=out$gradient;

hess1=out$hessian;

#second optimization method

library(numDeriv)

out <- nlminb(init\_p, MeisamGMMfunc)

print(out);

param=out$par;

hess1<- hessian(func=MeisamGMMfunc, x=out$par)

#third Optimization Method

# method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent"

#method that worked well: L-BFGS-B

#Method "L-BFGS-B" is that of Byrd et. al. (1995) which allows box constraints, that is each variable can be given a lower and/or upper bound. The initial value must satisfy the constraints. This uses a limited-memory modification of the BFGS quasi-Newton method. If non-trivial bounds are supplied, this method will be selected, with a warning.

out <- optim(init\_p, MeisamGMMfunc, method="L-BFGS-B",

control = list(maxit = 30000, temp = 2000, trace = TRUE,

REPORT = 500), hessian=T) #, fnscale=-1

param = out$par;

hess1= out$hessian;

fval = out$value;

#Fourth optimization Method: Genetic Algorithm

library(rgenoud)

out <- genoud(MeisamGMMfunc, nvars=7,max=FALSE) #didn't work

library(DEoptim)

lower <- c(-10,-10)

upper <- -lower

## run DEoptim and set a seed first for replicability

#set.seed(1234)

DEoptim(MeisamGMMfunc, lower, upper)

param = out$par;

hess1= out$hessian;

fval = out$value;

Rprof(NULL);

std = diag(ginv(hess1));

std = sqrt(std);

trat = cbind(t(-exp(param[1:2])),t(exp(param[3:4])))/t(std[1:4]);

cat('parm estimates and t-stat for (bp,ah,a, c, v) are: \n');

cat(cbind(t(-exp(param[1:2])),t(exp(param[3:4])),param[6]^2),'\n')

cat(cbind(t(trat[1:4]),(param(6)^2)/std[6]));

bp\_e = -exp(param[1]);

ah\_e = -exp(param[2]);

a\_e = exp(param[3]);

c\_e = exp(param[4]);

tt1\_e = param[5];

v\_e = param[6]^2;

betar\_e =tt1\_e/c\_e;

STEFOC=cbind(1/c\_e,-tt1\_e/(c\_e^2));

ParamCovar =hess1[c(4,5),c(4,5)]\*(2\*J);

betarSTE=sqrt(STEFOC%\*%ParamCovar%\*%t(STEFOC)/(2\*J));

cat('stock out regret coefficient and tstat is is: \n');

cat(betar\_e,'\n');

cat((betar\_e/betarSTE),'\n')

cat('Auto correlation coefficient is:\n');

cat(exp(param[7]),'\n')

cat((exp(param[7])/std[7]),'\n')

LL=-fval;

p=7;

AIC = 2\*p-2\*LL;

BIC = -2\*LL+p\*log(J);

cat('Log Likelihood, AIC, BIC is:\n');

cat(cbind(LL,AIC,BIC),'\n');

Without heterogeneity code of Regret pricing Project:

#modified code only high price regret and stock out regret, with modified

# specification of the stock out regret

#clear workspace

rm(list = ls());

Rprof("AggregLogitNoHeterogen.out")

J = 10000;

T = 3;

P1 = sample(1:20, J, replace=T); #generate random integer number

discount = runif(J, 0, 1); #generate uniform random number

P2 = ceiling(P1\*discount);

lambda = runif(J, 0, 1); #availability

xi = matrix(rnorm(20), J,2);

# alpha = 2;

# c = 0.5;

# bp = -0.2;

alpha = 2\*runif(1,0,1);

c = runif(1,0,1);

bp = runif(1,0,1);

gamma =.975; #discount factor

P = cbind(P1, P2);

Pn = matrix(P,nrow=J\*2);

# high price regret coefficient

alphap = 3\*runif(1,0,1);

# stock out regret coefficient

betar = 5\*runif(1,0,1);

# calculate utility

uij1 = alpha+(c+gamma\*c)+bp\*P1+ alphap\*lambda\*(P1-P2)+ xi[,1];

uij2 = gamma\*(lambda\*(alpha+c+bp\*P2)+ betar\*(rep(1,J)-lambda)\*(c+gamma\*c))+ xi[,2];

e1=exp(uij1); e2=exp(uij2);

shares=cbind(e1/(1+e1+e2),e2/(1+e1+e2));

outside=cbind(1./(1+e1+e2),1./(1+e1+e2));

shares\_n=matrix(t(shares),nrow=J\*2); #stack shares on top of eachother

outside\_n=matrix(t(outside),nrow=J\*2); #stack outside share

#data

rm(list = ls());

#global P1 P2 Dur1 Dur2 lambda gamma shares km cost %outside\_n

#global vcm se\_est betas variance

library(xlsx);

num<-read.xlsx("C:/Users/MHE/Desktop/ActiveCourses/Projects/BehavioralPricing/Data/cleaned10232013.xlsx",1)

#cbind(summary(num))

names(num)

J = dim(num)[1];

p = dim(num)[2];

Dur1=num[,2];

Dur2=num[,3];

P1=num[,4];

P2=num[,5];

Av1=num[,6];

Av2=num[,7];

Av3=num[,12];

S1=num[,8];

S2=num[,9];

MKTSz=num[,10];

cost = num[,11];

T = 3;

# test for cut off of second period

# P3=num[,13];

# P4=num[,16];

# Dur3=num[,14];

# S3=num[,15];

# S2=S3;

# P2=P4;

#Dur2=Dur3;

# test for average of second period availability, and normalized S2

lambda = Av2;

# use normalize availability

#lambda = Av2/Av1; #availability

#lambda = Av3; # availability at the beggining of second period

# test for normalized sales of second period

MKTSz=MKTSz/1.25;

cf=1.25;

MKTSz=MKTSz\*cf;

MKTSz = MKTSz - (cf\*S2);

S2 = S2/lambda;

MKTSz = MKTSz + (cf\*S2);

# tstat=regstats(P1,P1-P2,'linear'), tstat.rsquare=.711 => VIF=.711

#gamma =.975; #discount factor

gamma=1/(1+.0025)^Dur1;

# create shares

shares=cbind(S1/MKTSz,S2/MKTSz);

outside=matrix(rep(rep(1,J)-rowSums(shares),each=2), ncol=2, byrow=TRUE);

shares\_n=matrix(shares,nrow=J\*2); #stack shares on top of eachother

outside\_n=matrix(t(outside),nrow=J\*2); #stack outside share

#beta=cbind(alpha,c,bp);

# X1= cbind(rep(1,J),(1+gamma)\*rep(1,J),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda,gamma\*lambda,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(1+gamma));

# no fixed effect and heterogeneity

# p=5;

# X1= cbind(rep(1,J),(Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda,gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(Dur1+gamma\*Dur2));

#no fixed effect, but heterogeneity with market size and cost

# p=5;

# X1= cbind(rep(1,J)/MKTSz,(0.5\*Dur1+gamma\*Dur2)\*cost,P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda/MKTSz,0.5\*gamma\*lambda\*Dur2\*cost,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2)\*cost);

#no fixed effect, but heterogeneity with market size and cost (include

#cherry picking and fixed it to negative of regret coefficient)

# p=5;

# X1= cbind(rep(1,J)/MKTSz,(0.5\*Dur1+gamma\*Dur2)\*cost,P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda/MKTSz,0.5\*gamma\*lambda\*Dur2\*cost,gamma\*lambda\*P2,-gamma\*lambda\*(P1-P2),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2)\*cost);

#no fixed effect, but heterogeneity only in ownership through MKTSz (include %cherry picking as separate coefficient)

#p = 6;

#X1 = cbind(rep(0,J),rep(1,J)/MKTSz,(0.5\*Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

#X2 = cbind(gamma\*lambda\*(P1-P2),gamma\*lambda/MKTSz,0.5\*gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2));

#no fixed effect, but heterogeneity with market size and cost (include

#cherry picking as separate coefficient)

# p=6;

# X1= cbind(rep(0,J),rep(1,J)/MKTSz,(0.5\*Dur1+gamma\*Dur2)\*cost,P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda\*(P1-P2),gamma\*lambda/MKTSz,0.5\*gamma\*lambda\*Dur2\*cost,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2)\*cost);

# no fixed effect but capture effect of consumption heterogeneity value with cost

# capture hterogeneity in consumption utility using cost data

# p = 5;

# X1 = cbind(rep(1,J),(Dur1+gamma\*Dur2)\*cost,P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda,gamma\*lambda\*Dur2\*cost,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(Dur1+gamma\*Dur2)\*cost);

# include heterogeneity using cost both in consumption utility and

# ownership utility

# p=5;

# X1= cbind(rep(1,J)\*cost,(Dur1+gamma\*Dur2)\*cost,P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda\*cost,gamma\*lambda\*Dur2\*cost,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(Dur1+gamma\*Dur2)\*cost);

# inclusion of heterogeneity using cost only in stock out regret and

# ownership utility

# p=5;

# X1= cbind(rep(1,J)\*cost,(Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda\*cost,gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(Dur1+gamma\*Dur2)\*cost);

# fixed effect (duration 0.5 because it is average duration of usage)

# p=J+4;

# X1= cbind(diag(J),(0.5\*Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(diag(J)\*gamma\*lambda,0.5\*gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2));

# fixed effect with consumption utility heterogeneity inclusion

# p=J+4;

# X1= cbind(diag(J),(0.5\*Dur1+gamma\*Dur2)\*cost,P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(diag(J)\*gamma\*lambda,0.5\*gamma\*lambda\*Dur2\*cost,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2)\*cost);

# fixed effect with heterogeneity of consumption in regret, but not in

# consumption utility directly

# p=J+4;

# X1= cbind(diag(J),(0.5\*Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(diag(J)\*gamma\*lambda, 0.5\*gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(0.5\*Dur1+gamma\*Dur2)\*cost);

# fixed effect with markdown dummy

# p=J+5;

# X1= cbind(diag(J),rep(1,J),(Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(diag(J)\*gamma\*lambda,gamma\*lambda,gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(Dur1+gamma\*Dur2));

# including cost rather than fixed effect

# p=6;

# X1= cbind(rep(1,J),cost,(Dur1+gamma\*Dur2),P1,lambda\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda,gamma\*lambda\*cost,gamma\*lambda\*Dur2,gamma\*lambda\*P2,rep(0,J),gamma\*(rep(1,J)-lambda)\*(Dur1+gamma\*Dur2));

# fixed effect, introducing availability in the first period

# p=J+4;

# lambda1=Av1;

# lambda2=Av2;

# X1= cbind(diag(J)\*lambda1,lambda1\*(Dur1+gamma\*Dur2),lambda1\*P1,lambda2\*(P1-P2),rep(0,J));

# X2= cbind(diag(J)\*gamma\*lambda2,gamma\*lambda2\*Dur2,gamma\*lambda2\*P2,rep(0,J),gamma\*lambda1\*(Dur1+gamma.\*Dur2));

# heterogeneity with cost, introducing availability in the first period

# p=6;

# lambda1=Av1;

# lambda2=Av2;

# X1= cbind(lambda1,lambda\*cost,lambda1\*(Dur1+gamma\*Dur2),lambda1\*P1,lambda2\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda2,gamma\*lambda2\*cost,gamma\*lambda2\*Dur2,gamma\*lambda2\*P2,rep(0,J),gamma\*lambda1\*(Dur1+gamma\*Dur2));

# no fixed effect, introducing availability in the first period

# p=5;

# lambda1=Av1;

# lambda2=Av2;

# X1= cbind(lambda1,lambda1\*(Dur1+gamma\*Dur2),lambda1\*P1,lambda2\*(P1-P2),rep(0,J));

# X2= cbind(gamma\*lambda2,gamma\*lambda2\*Dur2,gamma\*lambda2\*P2,rep(0,J),gamma\*lambda1\*(Dur1+gamma\*Dur2));

X = t(cbind(X1,X2));

#Xn=matrix(X,ncol=J\*2,nrow=5);

Xn= matrix(X,ncol=J\*2, nrow=p);

Xn=t(Xn); #stack X's

Yn=log(shares\_n/outside\_n);

#OLS

betas=solve(t(Xn)%\*%Xn)%\*%t(Xn)%\*%Yn;

errors=Yn-Xn%\*%betas;

vcm=as.vector(t(errors)%\*%errors)\*solve(t(Xn)%\*%Xn)/(2\*J);

se\_est=sqrt(diag(vcm));

#params=[alpha c bp alphap betar];

betas = t(betas);

se\_est = t(se\_est);

a\_e = t(betas[1,1:(p-4)]);

c\_e = betas[1,(p-3)];

bp\_e = betas[1,(p-2)];

alphap\_e= betas[1,(p-1)];

tt1\_e = betas[1,p];

betar\_e =tt1\_e/c\_e;

STEFOC=cbind(1/c\_e,-tt1\_e/(c\_e^2));

#ParamCovar =vcm[c(2,5),c(2,5)]\*J;

ParamCovar =vcm[c(p-3,p),c(p-3,p)]\*(2\*J);

betarSTE=sqrt(STEFOC%\*%ParamCovar%\*%t(STEFOC)/(2\*J));

#rbind(cbind(params,betas[1,1:4],betar\_e),cbind(betas[1,1:4]/se\_est[1,1:4],betar\_e/betarSTE));

#rbind(cbind(betas[1,1:4],betar\_e),cbind(betas[1,1:4]/se\_est[1,1:4],betar\_e/betarSTE));

# c bp alphap betar

variance = colMeans(errors^2)\*2\*J/(2\*J-1);

LL = - 2\*J\*log(sqrt(2\*pi\*variance)) - .5\*sum(errors^2/variance);

AIC = 2\*p-2\*LL;

BIC = -2\*LL+p\*log(J);

cat('Log Likelihood, AIC, BIC is:\n');

cat(cbind(LL,AIC,BIC),'\n');

cat('estimation and t-stat for direct estimation for (c,bp,alphap,betar) is:\n');

cat(cbind(t(betas[1,(p-3):(p-1)]),betar\_e), '\n');

cat(cbind(t(se\_est[1,(p-3):(p-1)]),betarSTE),'\n');

cat(cbind(t(betas[1,(p-3):(p-1)]/se\_est[1,(p-3):(p-1)]),betar\_e/betarSTE),'\n');

# intercept, single intercept

cat('Intercept is: \n');

cat(betas[1,(p-4)],'\n');

cat(se\_est[1,(p-4)],'\n');

cat(betas[1,(p-4)]/se\_est[1,(p-4)],'\n');

Rprof(NULL);

# for fixed effects

cat('Fixed Effects are: \n');

cat(betas[1,1:(p-4)], '\n');

cat(se\_est[1,1:(p-4)],'\n');

cat(betas[1,1:(p-4)]/se\_est[1,1:(p-4)],'\n');